UNIVERSITY OF ALBERTA

TOPOLOGY DESIGN OF MESH-RESTORABLE NETWORKS

by

©

CHIOMA ANTHONETTA EZEMA

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of
the requirements for the degree of Master of Science

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Edmonton, Alberta

Spring 2003

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-82273-7

Canada

University of Alberta

Library Release Form

Name of Author: Chioma Anthonetta Ezema

Title of Thesis: Topology Design of Mesh-Restorable Networks

Degree: Master of Science

Year this Degree Granted: 2003

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

12827 102 Street,
Edmonton, AB T5E 4J2
Canada

Date: 6th Jan. 2003

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of graduate Studies and research for acceptance, a thesis entitled

"Topology Design of Mesh-Restorable Networks"

submitted by

Chioma Anthonetta Ezema

in partial fulfillment of the requirements for the degree of **Master of Science**.

WDGrover

Wayne D. Grover (Supervisor)

Bruce Cockburn (Internal)

M. MacGregor

Mike MacGregor (External)

Date: Jan 6/03

ABSTRACT

The Mesh Topology, Routing and Sparing (MTRS) problem involves the simultaneous optimization of topology, working and spare capacity requirements for a mesh-restorable network. Combinatorial principles show that on an *n*-node graph, there are vastly more arrangements of closed connected graphs with many edges than there are graphs that have relatively few edges that describe a connected closed graph. Thus the computational complexity of choosing the optimal topology from the set of all possible edges of the *n*-node graph, along with the capacity design renders IP methods infeasible for solving the complete problem.

In the thesis, we propose and evaluate edge-limiting heuristics that constrain the solution space considered for the problem to a space containing highly qualified and plausible edges and several heuristic strategies that take the topology search sub-problem away from MIP solver domain. Qualified topologies found from the heuristics are then passed to a MIP solver for further optimization.

ACKNOWLEDGEMENT

In carrying out my studies here at the University of Alberta, a number of people and organizations have assisted me, been there for me and most importantly, encouraged me in my study. With a few lines, I would like to pay a tribute to them:

My husband Nd, my parents Paul and Grace and my in-laws especially Ify and Emilia for their love and support.

Special thanks to my supervisor Dr. Wayne D. Grover for his guidance and encouragement over the course of this work, as well as his painstaking efforts to provide timely and detailed markups to the thesis draft and providing numerous key ideas that were used throughout this research. Indeed, I owe the successful completion of this work entirely to him.

The invaluable sources of funds that I had available to me: The Canadian Commonwealth and Scholarship Program that gave me the opportunity to study in Canada, Dr. Wayne Grover and TR*Labs* Edmonton.

I would also like to thank specially Ofer Shai and John Doucette for their inputs and assistance in carrying out my work. Dr. Mike MacGregor for his support and kindness especially during the first year of my study.

I would like to thank the Staff and students (especially members of the Network Systems Research Group led by Dr. Grover) of TR*Labs* Edmonton for their tremendous support and for providing a conducive environment in which to carry out my studies.

The friends I made in Canada who opened up their families to me and made me feel very much at home – Adamma, Chimezie and Chinyere.

Finally, much thanks to my bosom friends Efe and Gift for being there for me on this journey.

Dedicated to my husband Ndubuisi

and to my parents Paul and Grace

TABLE OF CONTENTS

1. TELECOMMUNICATIONS TRANSPORT NETWORKS..1

1.1 INTRODUCTION..1

1.2 TERMINOLOGY AND BASIC CONCEPTS..3

1.3 LITERATURE REVIEW..5

1.4 THESIS GOAL..9

1.5 THESIS OUTLINE..12

2. MESH NETWORK CARRIER FACILITY DESIGN..14

2.1 INTRODUCTION..14

2.2 CAPACITY DESIGN..15

2.3 THE EFFECT OF TOPOLOGY ON NETWORK COSTS..19

2.4 IMPORTANCE OF EDGE (SPAN) COSTS..23

2.5 PRIOR OPTIMIZATION PROBLEMS..25

2.5.1 Transhipment or Flow assignement: A Basic Choice..26

2.5.2 Choice of Transhipment Model For Solving MTRS Problems..36

2.6 A LOOK AT SOME REAL NETWORKS..38

2.7 ELEMENTAL GRAPH THEORY AND ALGORITHMS..43

3. MESH TOPOLOGY, ROUTING AND SPARING (MTRS)..54

3.1 INTRODUCTION..54

3.2 THE COMPLEXITY OF THE COMPLETE MTRS PROBLEM..59

4. GRAPH SIFTER METHODOLOGY AND ARCHITECTURE..64

4.1 GRAPH GENERATION..65

4.1.1 Sequential Graph Generation Method..66

4.1.2 Random Generation Methods..68

4.1.3 Algorithmic Graph Generation Methods..71

4.2 GRAPH TESTING..74

4.2.1 Test 1 - Test For Minimum And Maximum Edge Count..74

4.2.2 Test 2 - Test for Stub Nodes..75

4.2.3 Test 3 - Test For a Connected Graph..75

4.2.4 Test 4 - Test for Minimum Cost graph..76

4.2.5 Test 5 - Test for Two-edge Connected Graph..76

4.2.6 Test 6 - Test for Bi-connected Graph..76

5. PRELIMINARY STUDIES 83

5.1 Computing Environment 83

5.2 Statistical Studies Using The Graph Sifting Approach 84

5.3 Empirical Studies Of The Effect Of Ω On Total Network Costs 89

5.4 The Effect of Demand Intensity And Pattern On Network Costs 97

6. LOCAL SEARCH HEURISTICS: IMPROVING ON THE GRAPH SIFTER 109

6.1 Introduction 109

6.2 A priori Limitation Of The Candidate Edge Set 110

6.3 Sweep Search Heuristic 123

6.4 Iterative Sampling Heuristic 130

6.5 Complexity Analysis Of The Heuristics 137

7. TESTING AND RESULTS 140

7.1 Introduction 140

7.2 Experimental Results 142

7.2.1. Iterative Sampling heuristic results 155

7.2.2 Sweep Search Results 157

7.3 A comparison of Our heuristic results with 3-Step heuristic results 160

8. APPLICATION TO INCREMENTAL GROWTH PLANNING 162

8.1 Scenario 1: Extending the reach of the network 164

8.2 Scenario 2: Increasing the connectivity of the network 169

9. CONCLUSION 173

9.1 Summary Of Thesis 173

9.2 Directions for Further Studies 176

REFERENCES 179

APPENDIX A 185

APPENDIX B 199

APPENDIX C 215

LIST OF TABLES

Table 1 - Eligible working routes for network 5n7s 30

Table 2 – Eligible restoration routes for network 5n7s 31

Table 3 – Variables and Constraints of the MTRS problem for some node sizes 60

Table 4 – Time complexity with network size 67

Table 5 - Test Suite Results (universe of all possible edges used) 85

Table 6 – Test Suite Results (subset of possible edges used) 85

Table 7 –Test results for a sample of random graphs on larger N 88

Table 8 – Edge Utilization 94

Table 9 – Statistical properties of the demand patterns 100

Table 10 – Magnitude of reduction for the edge set (for $d_{\max} = 5$) 112

Table 11 – Edge limiting solutions (MTRS) 113

Table 12 – Solution after edge limiting heuristic 114

Table 13 – Reduction for $k = 3$ and $m = 0.2N(N-1)$ 118

Table 14 - Reduction for $k = \{3,4\}$ and $m = 0.3\times|N|.d_{\max}$ 120

Table 15 – Size of candidate edge sets used for testing ($m = 0.3\times|N|.d_{\max}$, $k = 3$) 142

Table 16 – Complete and reduced edge set MTRS solutions for test series 1 problems 143

Table 17 – Reduced MTRS vs. Iterative Sampling results 146

Table 18 – Iterative Sampling heuristic test results 152

Table 19 – MTRS reference results for test series 2 154

Table 20 – Iterative Sampling heuristic results 155

Table 21 – Progress of the Sweep Search heuristic for the 15n59s network 158

LIST OF FIGURES

Figure 1 - Logical vs. Physical view of a network 3

Figure 2 – Span Restoration/Protection vs. Path Restoration/Protection for a single path 17

Figure 3 – Multiple restoration routes for a single span failure 18

Figure 4 – The capacity design problem 18

Figure 5 – The topology and capacity design problem 19

Figure 6 – Fixed topology design problem 20

Figure 7 – Variable topology design problem 21

Figure 8 - Networks Costs vs. Connectivity 22

Figure 9 – Model mesh-restorable network topology (Network 5n7s) 27

Figure 10 – Level 3 Fiber Network for the United States 40

Figure 11 – Genuity Fiber Optic Network for the United States 41

Figure 12 – The JANET backbone network for the United Kingdom 41

Figure 13 – The Belgian Transport network 42

Figure 14 – COST239 European Model Network 42

Figure 15 – MIP solutions for 9-node network 61

Figure 16 - Architecture of the proposed graph sifter approach 65

Figure 17 - Two Trees graph generation 73

Figure 18 – Methods for graph generation 74

Figure 19 – The Graph Sifter 78

Figure 20 – The Graph Sifter coupled with the optional repair process 81

Figure 21 – Flow diagram illustrating the composite graph generation and sifting modules 82

Figure 22 – Summary of graph qualifications with generation given complete edge set 87

Figure 23 – Graph qualifications with generation given subset of universe of edges 87

Figure 24 – Total cost of optimal MTRS solution vs. Ω for network COST239-8n 92

Figure 25 - Total cost of optimal MTRS solution vs. degree for network COST239-8n 92

Figure 26 – Optimal Solution for $\Omega = 1$ for COST239-8n....95

Figure 27 – Optimal Solution for $\Omega = 5, 6, 7$ for COST239-8n....95

Figure 28 – Optimal Solution for $\Omega = 8, 9, 10, 15$ for COST239-8n....95

Figure 29 – Optimal Solution for $\Omega = 20, 25, 30, 35, 40$ for COST239-8n....96

Figure 30 – Optimal Solution for $\Omega = 45, 50, 55, 60$ for COST239-8n....96

Figure 31 – Optimal Solution for $\Omega = 65, 70, 75, 80, 85, 90, 95, 100$ for COST239-8n....96

Figure 32 – Demand distribution between O-D pairs for three demand patterns....99

Figure 33 - Cost vs. degree for C239 demand pattern.... 101

Figure 34 – Cost vs. degree for C239 (x 2) demand pattern.... 102

Figure 35 - Cost vs. degree for C239 (x 5) demand pattern.... 102

Figure 36 - Cost vs. degree for C239 (x 10) demand pattern.... 103

Figure 37 - Cost vs. degree for Random demand intensity.... 103

Figure 38 - Cost vs. degree for GR50 demand intensity.... 104

Figure 39 – Optimal solutions for $\Omega = 20, 50, 100$ with varying demand patterns.... 105

Figure 40 - Edge limitation with various *(m, k)* parameters for network 7n21s2.... 121

Figure 41 – Reduction methods 4 and 5 on a larger network (15n59s).... 122

Figure 42 - Oscillating Sweep Search strategy on topology.... 125

Figure 43 – The PARE-UP/PARE-DOWN routine.... 129

Figure 44 – The Sweep routine.... 129

Figure 45 – The Sweep Search flow diagram.... 130

Figure 46 – Convergence on solution for iterative sampling.... 134

Figure 47 – Topology changes with edge additions (1).... 135

Figure 48 – Topology changes with edge additions (2).... 136

Figure 49 – The Iterative Sampling flow diagram.... 137

Figure 50 – Initial search of 2 sweeps from d_{max} to d_{min} for 7n21s2-G.... 148

Figure 51 – Narrowed search of 4 sweeps from d_{max} to d_{min} for 7n21s2-G 149

Figure 52 – Ideal performance of criteria for narrowing search region 150

Figure 53 - Initial search of 2 sweeps from d_{max} to d_{min} for 7n21s2-G 150

Figure 54 – Average cost versus nodal degree 151

Figure 55 - Narrowed search of 4 sweeps from d_{max} to d_{min} for 7n21s2-G 151

Figure 56 - Rate of growth of run time of Iterative sampling with problem size 157

Figure 57 – Sweep Searching for network 15n59s 159

Figure 58 – The topology problem of growing a 6-node network to a 9-node network 165

1. TELECOMMUNICATIONS TRANSPORT NETWORKS

1.1 INTRODUCTION

A transport network is a set of physical facility routes on which transmission and switching facilities provide point-to-point transmission paths for routing demands. Such a network consists of the switching devices (nodes) and the set of media (fiber, microwave radio) that connect these nodes together (spans). In the early transport networks, restoration of demands in the event of a failure (survivability) was not much of an issue. Consequently, network topologies were essentially tree-like since the aim was to minimize the cost of serving the working demands, without any restorability targets.

Today, such topologies are not an option. Our dependence on communications systems means that failure of the network without restoration for as little as a few seconds could cause great loss and have severe consequences. The network topology in use today, has to achieve certain minimum restorability objectives such as restoration of 100% (or some target level for the class of service) of working demands in event of the failure of any network component.

The wide use of fiber optics as cable medium stresses the need for network designs that meet certain restorability objectives because the cable-based medium has exhibited a lower structural availability than microwave radio, the latter being predominantly used in early transport networks. Indeed, [VePo02] has it that "in existing TDM and data networks, reliability in a metropolitan DWDM network is a major concern to service providers. This concern is heightened by the FCC statistics indicating metro networks experience 13 fiber cuts per 1000 miles per year (compared to 3 fiber cuts per 1000 miles per year for long-haul networks)". The shift today is to mesh-restorable networks, which are accepted to be a cost-effective solution in situations where network demand and network connectivity are high [PoDe97].

The term *mesh* can be used to describe connectivity at different layers in the network from traffic pattern to fiber topology. The fiber topology for a network can be described as mesh if there is a sufficient amount of diversely routed fiber interconnecting the nodes. A traffic pattern can also be described as mesh if there are no major hubs and demands are distributed between node pairs [CGKW02]. Full mesh connectivity describes the situation where there exists a span or demand quantity between every node pair. The graph of the transport network in this respect is not a full mesh, but an irregular mesh where the fiber connectivity is not complete between every node pair.

A mesh network is one that has a mesh fiber topology and a mesh-restorable network design is one that supports mesh restoration and protection schemes (spare capacity is placed on one span to contribute to the restorability of some other failed span(s)). The term 'mesh-restorable' is used to reflect the ability of a routing or restoration mechanism to exploit the mesh fiber topology [GrDo01]. In this thesis work, a mesh topology refers to a physical topology in which a network node can connect to more than two other network nodes.

Figure 1 shows the logical and physical views of a network. The logical view represents demand pairs in the network and therefore, the paths that are provisioned within the network. The physical view shows the actual fiber deployment for the network, over which the logical topology is realized. It is important to note at this point that the average nodal degree for a network as used in this project is derived from the physical view of the network and not the logical view.

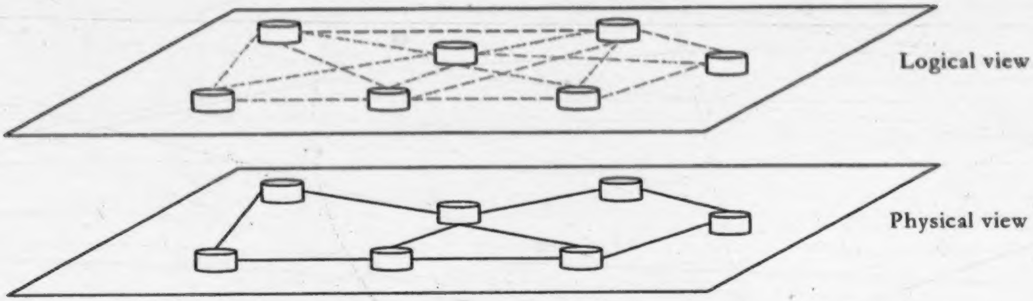


Figure 1 - Logical vs. Physical view of a network

In a mesh-restorable transport network, spare capacity is designed into the spans between nodes, allowing them to take on the additional load of re-routed payload signals (restoration path sets) that result from a failure. This ensures the survival of these signals in the event of a cable cut. In ring networks and appropriately designed mesh networks, we are also able to protect transiting traffic from node loss.

Sharing of spare capacity is the attractive economic feature of a mesh-restorable network and the extent to which this is achievable depends on the underlying topology. The higher the degree of capacity sharing there is, the lower the spare (and working) capacity placed and consequently, the more economical the network. A closed high-degree topology is therefore, desirable for the efficient sharing of the spare capacity allocations over non-simultaneous failure scenarios (where a closed topology refers to a graph with at least two edge-disjoint or two-node disjoint paths between every node pair).

1.2 TERMINOLOGY AND BASIC CONCEPTS

In this section, we introduce some terminology, acronyms and basic concepts that would be used throughout this work.

Average nodal degree	The degree of a node in a graph is the number of edges that are incident on the node. When considering the entire nodes in a graph, the average

	nodal degree is then a measure of the average number of edges incident on each node.
Candidate edge set	The candidate edge set is the set of edges that are considered in a particular problem instance for inclusion in the solution to the problem of finding the optimal topology.
Connectivity	The term graph connectivity is used interchangeably with graph average nodal degree and has the same meaning.
Edge	When we are referring only to the topology of a network, we use the term edge to mean the connection or link between nodes.
Feasible topology	A feasible topology for a transport network is one that satisfies the survivability and connectivity constraints we impose.
MTRS	MTRS stands for Mesh Topology, Routing and Sparing problem where we consider the selection of an optimal topology from the candidate edge set comprising of all possible edges on V vertices $\left(m = V\left(V-1\right)/2\right)$.
Network	A network includes the topology, routing solutions and capacity assignments on each of its spans.
Qualified graph	A qualified graph (or topology) is a feasible topology i.e. satisfies all connectivity and survivability constraints.
Reduced or Restricted MTRS	The reduced or restricted MTRS problem is when we attempt to solve the MTRS problem on a subset of its universal edge set $\left(m' < V\left(V-1\right)/2\right)$.
Span	When we are considering the composite network solution (topology and capacity), we use the term span to mean the connection or link between nodes.
Stub nodes	A stub node is a node with a degree of 1.
Survivability	Survivability refers to the ability of a network to continue functioning without disruption in the event of a physical failure of any of its components. A network that satisfies this condition is termed a survivable network. To be survivable, first, the network topology has to be two-connected.

	Second, capacity has to be provisioned in the network to support the demand flows that are affected by the failure.
Topology	The topology of a network is a graph that describes the physical layout of its nodes and spans.
Weight	The weight of a graph is the number of edges the graph has.

1.3 LITERATURE REVIEW

There are a vast number of studies that deal with topological design of communication networks. Some examples of issues that have been treated in many of these studies include: Internet topologies ([ZeCB96, ZeCD97, MeMB00), network access planning ([Gavi91]), expansion planning ([BaMW95]), and backbone network design ([Kers93, Robe99]) among others.

The spare capacity assignment problem in mesh-restorable networks has been treated in a number of studies including [GrBV91, Herz93, HeBU95, GrDo01, DoGr02a, DoGr02b]. Given a network topology and the amount of working capacity placed on each span in the network, these studies address the problem of allocating the minimum amount of capacity on each span to ensure the survivability of the network for different restoration mechanisms and failure scenarios.

There are also a number of studies dealing with the design of minimum-cost survivable networks such as [MoSh89], [MoMP90]. In [MoMP90], the authors consider the problem of designing a minimum cost network subject to certain two-connected survivability constraints on nodes designated as special nodes. The work deals only with the topology design of the network without the associated capacity design problem.

In [MoSh89], the authors present a structural characterization of optimal solutions to the problem of designing a minimum-weight two-connected network spanning a set of vertices

V for which there is a symmetric nonnegative distance function $d(\cdot)$ defined on $V \times V$ which satisfies the triangle inequality. The authors propose that there exists an optimal two-connected solution whose vertices all have degree 2 or 3, such that the removal of any edge or pair of edges leaves a bridge in the network. Again, this work studies only the physical design of a minimum cost network. As we will show, a consideration of capacity design along with the topology design often yields optimal minimum cost network solutions in which there exists vertices that have degrees greater than 3.

[RMRR01] develops a bi-criteria approximation algorithm for solving the 'degree-constrained minimum-cost network' problem. This problem has two objectives: (1) minimizing the degree of the network (there is a maximum nodal degree for any of the vertices in the network) (2) minimizing the total cost of the network. There is no survivability requirement to this network and the result is a Steiner tree.

In contrast, there are relatively few studies that deal with the problem of mesh-restorable network topology design in conjunction with routing and working capacity design and the restoration capacity requirements of such networks. Through the literature, we find relatively few studies [Mino81, GaTD89, Kers93, StDa94, PoDe97, PPLD97, PiDe00, GrDo01] where the topology of a survivable network is considered in conjunction with the routing and capacity requirements for the network.

Routing in a network would require finding paths between all pairs of nodes. This is in itself a procedure of complexity $O(N^3)$. There are $O(N^2)$ candidate edges to consider in selecting a mesh topology. This pushes the complexity of algorithms that deal with mesh topology and capacity design problems towards $O(N^5)$. As N gets larger, the problem gets computationally prohibitive. As a result, abstractions of the problem are often employed when providing fast and approximate solutions to parts of the complete problem, especially involving the routing and capacity design sub-problem.

The work of [Mino81, GaTD89, StDa94] concerns the design of survivable networks that allows for the entire rerouting of all demands in the network (not only the affected demands) in the event of a network component failure. Thus, they assume that there is no relationship between routing in the normal (without failure) state and routing in each of the failed states in the network (each failed state is defined by the set of network components that failed) because there is no restriction on the reconfiguration of the network after a failure occurs. Based on their assumption, the authors develop formulations that are essentially invalid when considering restoration schemes used in the mesh-restorable network in which only demands affected by a network failure are reconfigured (re-routed) after a failure.

[Kers93] discusses an $O\left(N^2\right)$ algorithm called MENTOR (MEsh Network Topology Optimization and Routing). This algorithm achieves its low complexity by replacing the actual detailed routing solution in its iterations with an easily computed surrogate criterion. This allows the MENTOR algorithm to concentrate on finding candidate topologies that have desirable network characteristics. The MENTOR algorithm was geared towards designing data network topologies and does not include any explicit consideration of network survivability.

[PoDe97] presents a mixed integer linear programming (MIP) formulation to the problem of designing the topology of a mesh-restorable network, and loading facilities (capacity systems) on the topology edges to satisfy the demand at a minimum cost while enabling a certain level of survivability with respect to link failures.

[PPLD97] presents a heuristic developed using a Genetic Algorithm for solving the near-optimal design of mesh-based span restorable networks i.e. determining the network topology and assigning capacity for serving demands and providing restoration in event of a single network failure. Here also, we find the use of surrogate methods for the routing and capacity design problem instead of detailed optimal capacity design (which is used only in evaluating the final best topology). The use of the surrogate routing and capacity allocation method over a detailed optimization solution provides for a faster means for evaluating the cost of candidate topologies thereby allowing more topology choices to be examined in the search.

In [PiDe00], a heuristic is presented that builds on the Genetic Algorithm (GA) heuristic from [PPLD97]. The GA provides a basic topology and an approximate capacity assignment design using the surrogate capacity allocation procedure. The quality of the topology is improved in a second phase using local improvement techniques: removing an edge, adding an edge or exchanging an edge with another. A third phase involves an exact capacity allocation procedure with further local optimization. A final phase leaves the topology unchanged and refines the working and spare capacity assignment.

In [GrDo01], the authors present a heuristic approach which uses MIP tools without explicit algorithmic search. The approach is to decompose the topology, routing and sparing problems into sub-problems each of which would be solved using the MIP solver. Insights gathered from the structure of the complete problem and attempts to limit the solution space that the solver considers led to this heuristic. The basic hypothesis is that the optimal edge set for the network design would be a subset of the union of edges that are considered optimal for routing demands and edges that provide restoration. The heuristic has 3 steps:

1. Step W1 - Solve a Fixed Charge plus Routing (FCR) problem (with no requirement for survivability) to identify edges that are sufficient for routing only. This is the problem of minimum cost of edge selections plus routing costs. The method tends to result in tree-like graph solutions.
2. Step S2 - Solve for the minimum cost of edges and capacity that would be added to the graph from step W1 to make the working flows restorable.
3. Step J3 - Solve the MTRS problem for the optimal topology design within the candidate edge set comprising of the union of edges from steps W1 and S2, rather than the universal edge set.

The purpose of steps W1 and S2 is to identify edge candidates from the set of all edges that would be good and viable for routing and restoration purposes individually. This has the effect of producing a limited solution space to be considered by the solver in step J3. Thus, the solver is guided into a space where only potentially good candidates are included in the problem. The heuristic solutions were reported to be within 8% of the optimal solution (where the latter was available for reference) and obtained in far less time. In problem instances where an optimal reference solution was not obtained, the heuristic solution obtained within 30 to 60 minutes was found to be 50% to 150% better than the solution to the full problem obtained within 6 to 12 hours.

1.4 THESIS GOAL

In this thesis, we study the combined problem of topology design - where to install the cables – and capacity assignment - how much capacity to provide on each span. For the rest of this work, we will refer to the combined problem of a carrier facility design as the Mesh Topology, Routing and Sparing (MTRS) problem [GrDo01]. We will further assume that the set of candidate edges is the universal set of $N(N-1)/2$ possible edges on N nodes except where stated otherwise.

It is possible to present the MTRS problem as a mixed integer programming (MIP) formulation and solve it with known linear programming methods. In [PoDe97], the authors develop a MIP formulation for the problem. Their observation was that the number of MIP branch and bound nodes that have to be evaluated for a small problem instance (8 nodes, 13 spans, 13 demands) "leaves no hope of solving larger problem instances to optimality". In [GrDo01], we understand that when the MIP solver is given an upper bound value for the network cost (obtained from the 3-step heuristic), it fails to find a feasible solution that improves on the heuristic within 6-18 hours for the networks tested. Without the bound, the solver is still searching in high connectivity topology space (leading to higher costs) as evidenced by the best solution found within the same time frame.

Given the level of complexity of the MTRS problem, the use of heuristic approaches seems to provide a way of obtaining near-optimal or approximating solutions. This thesis develops some heuristic approaches to solving the problem. The main idea we develop is to remove the topology search part of the problem out of the MIP domain by conducting a targeted and exhaustive search over light weight topology space by a separate means and then feeding the resulting plausible topologies to the MIP solver for the capacity assignment.

In the first part of our work, we attempt to randomly generate topologies that, if qualified, are fed to the capacity design solver. Test results with this approach (shown in Chapter 5.2) show that an algorithmic search method was necessary (the probability of randomly generating a qualified topology reduces as network size increases).

The second part of this work then presents two algorithmic topology search heuristics. The heuristics are based on local search techniques:

- identification of a starting solution x,
- local search in the neighborhood of x, $N(x)$ (solutions with one or more links added, removed or exchanged) while maintaining the minimum survivability constraints until no further improvement can be achieved.

This is similar to the approach that prior heuristics developed for solving the MTRS problems have taken (see [PPLD97], [PiDe00], [MoSh89]). The interesting differences come from the methods we employ for identifying the starting solution and choosing the neighborhood:

a. The first is the a priori limitation of the search space. Edges that are considered infeasible are removed from further consideration while ensuring that no edge $e \in E_{opt}$ is removed (where E_{opt} is the set of edges that comprise the optimal solution).

b. The second is the selection of a feasible starting topology. This is based on the minimum-cost spanning tree for the network. Subsequently, local optimization (adding an edge, removing an edge, exchanging edges) is carried out until no further improvement is achieved.

We apply these search heuristics in generating qualified candidate edge sets for the transport network. Each edge set can be used in either of two ways:

1. as an exact representation of a qualified graph. In this case, the topology is assumed to be fixed and further optimization involves only the capacity design for the given topology.
2. as the reduced candidate edge set. In this case, the edge specification lists edges that could potentially be included in the graph and further optimization involves solving the restricted instance of MTRS with the edge set as the candidate edge set.

1.5 THESIS OUTLINE

Chapter 2 deals with background on the mesh carrier facilities design problem. It discusses the effect of topology on the network cost, capacity design and presents some graph theoretic principles that are used in this work.

Chapter 3 introduces the Mesh Topology, Routing and Sparing (MTRS) problem and discusses the complexity of the complete problem.

Chapter 4 describes the architecture of the graph sifter.

Chapter 5 presents the results of tests carried out which defined the actual thrust of this thesis work. These tests include the sequential generation of graphs, a study of the effects of demand variation and edge-to-capacity cost ratio on network topology (and hence, on network costs).

Chapter 6 describes two local search heuristics – Sweep Search and Iterative Sampling – developed in this work. An analysis of their complexity follows.

Chapter 7 presents the results we obtained on "green-field" problem (where no edges are assumed to exist) with the heuristics and compare them to optimal solutions from a solver given the full problem.

Chapter 8 presents an application of the work in network topology evolution planning. We study the problem of extending the methods and results obtained in the green-field problem to the problem of extending the reach or connectivity of a 'legacy' network (where some edges are assumed to exist already, as a legacy from either a transport company, or utilities company).

Chapter 9 provides the summary and suggests directions for future work.

Appendices A to C document further data relevant to, and used in this thesis:

Appendix A has topology data for test networks used in this work.

Appendix B has demand data used.

Appendix C shows the AMPL models used.

2. MESH NETWORK CARRIER FACILITY DESIGN

2.1 INTRODUCTION

The design of a network is comprised of three sub-problems:

1. Design the fiber topology layout
2. Design the routing pattern and assign sufficient working capacity on each span and
3. Assign sufficient spare capacity on the spans to ensure full restorability for single component failures (a component is either a node or a span).

The solution to each of these sub-problems is dependent on the solution to the other sub-problems. The nature of this interdependence is as follows:

- the topology choice influences the routing of working traffic and hence, the cost of provisioning working capacity in the network
- the routing of working traffic influences the routing of restoration flows and hence the cost of provisioning spare capacity in the network
- topology affects the extent of sharing of spare capacity for restoration.

In [PiDe00], the concepts of sequential and integrated approaches for solving the MTRS problem are discussed. The sequential approach is in three phases with each of the sub-problems outlined above as a phase. Its advantage is that minimal resources are required (fast computation times, low memory requirement) since the problem is divided into easier to solve problems. However, the synergy (or interdependence) between the three components is lost and can lead to low quality or sub-optimal solutions (high cost networks).

On the other hand, the integrated approach attempts to simultaneously solve the three sub-problems. This approach makes it possible for high quality solutions (lower cost) to be realized because the interdependence of the sub-problems is exploited. However, the high resource requirement and computational complexity involved make this approach feasible only for very small networks.

A practical and effective method for solving the carrier facility design problem would involve a compromise between the sequential and integrated design approaches. The goal of this thesis is to develop such a method. We want to be able to carry out a targeted search of the solution space in a sequential manner, yet embodying the integrated design approach. Given:

1. the demand matrix (set of node-to-node demand quantities),
2. the set of candidate links with the distance matrix (distance of each link),
3. the cost of placing a unit of capacity on each link (proportional to the length of the link),
4. the cost of establishing each link (also proportional to the length of the link),

the problem we study is to determine which links to install, how to route the demands and in the event of a single link failure, how to restore the demand affected such that total network cost is minimized.

The next section discusses the traditional problem of capacity design.

2.2 CAPACITY DESIGN

Mesh-restorable networks are designed to provide real-time recovery of signals impacted by span cuts and/or node loss. To achieve this, spare capacity is pre-designed and pre-allocated on spans such that the surviving spans (spans not affected by the cut) can carry the payload from the impacted spans. When such provisions have been made for the restoration or protection of demands in the network against failures, we can say the network is *survivable*.

The approach to survivable network capacity design may be broadly classified into two categories:

- *Joint Capacity Placement (JCP)* – considers the simultaneous optimization of the minimum amounts of working and spare capacity needed for routing demands and for restoration in event of network element failure and
- *Spare Capacity Placement (SCP)* - the demands are first routed (typically shortest path) through the network, and then optimization of the minimum spare capacity needed to support restoration is carried out.

With the JCP design approach, there is an optimal co-ordination of the choice of routes taken for serving demands and providing restoration such that the total working and spare capacity requirements of the network is minimized. We can therefore expect that JCP would produce a more efficient capacity design (though some demands may not be shortest-path routed) but may be at the expense of increase in execution time. JCP would take longer to solve because the degree of complexity and number of constraints involved increases. With either JCP or SCP, a number of restoration or protection schemes are available.

Restoration and Protection, what is the difference? Protection occurs if for each demand pair, the working route(s) have pre-determined backup route(s) that are used when any network element failure affects the each working route. On the other hand, restoration occurs if there is an adaptive or real-time computation of paths from the available spare capacity in the network when any element failure occurs. Restoration expenses are adaptive to where the specific failure arises and so are generally more efficient.

The restoration and protections schemes can be classified by a number of attributes including:

- *Dedicated vs. Shared*: The bandwidth allotted for restoration or protection could be dedicated to specific working paths, or shared between a number of working paths that do not have network elements in common.
- *Path vs. Span*: For a span cut, the affected paths can be restored by rerouting the whole path between the origin and source nodes (path restoration/protection) or rerouting between the end nodes of the failed span (span restoration/protection). The span and path protection/restoration schemes are illustrated in Figure 2 for a single working path.

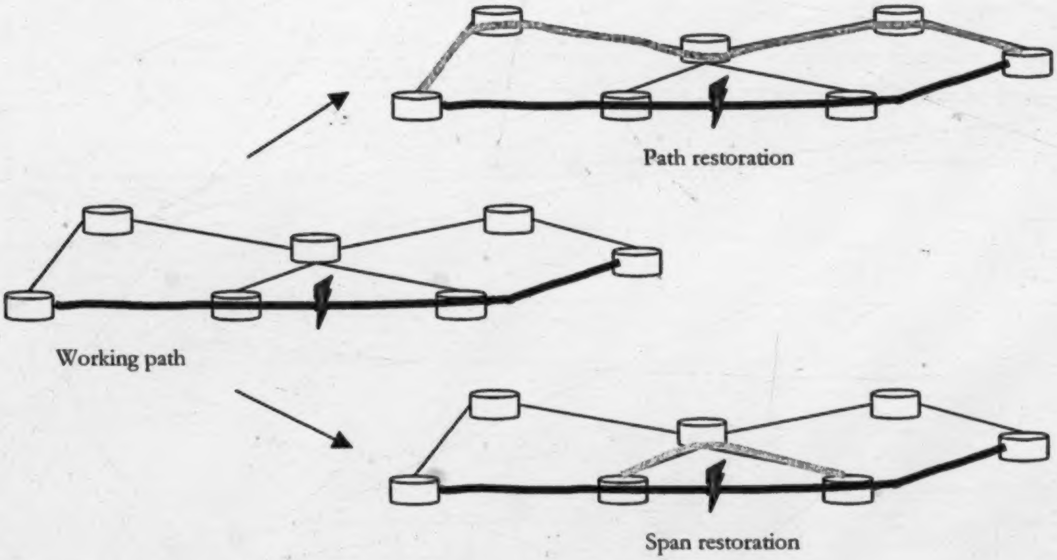


Figure 2 – Span Restoration/Protection vs. Path Restoration/Protection for a single path

The black flow lines show the working path prior to the span failure, the gray flow lines show the restoration path taken. When a failure occurs, many working paths are simultaneously affected. They all do not necessarily take the one single re-route shown in this example. A more typical span restoration re-route scenario is shown in Figure 3:

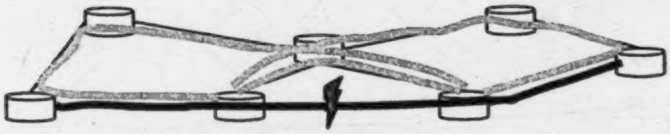


Figure 3 – Multiple restoration routes for a single span failure

In this work, the restoration mechanism employed is span restoration. For any failed span, restoration occurs through a set of replacement paths between the end nodes of the span by using the surviving spare capacity of spans in the network excluding the failed span. Demands remain on their previous routes up to and following the failure and demands that are not affected by the failure are not re-arranged.

The interaction of network topology and capacity design is best illustrated in the figures below. Figure 4 shows a summary of the capacity design problem.

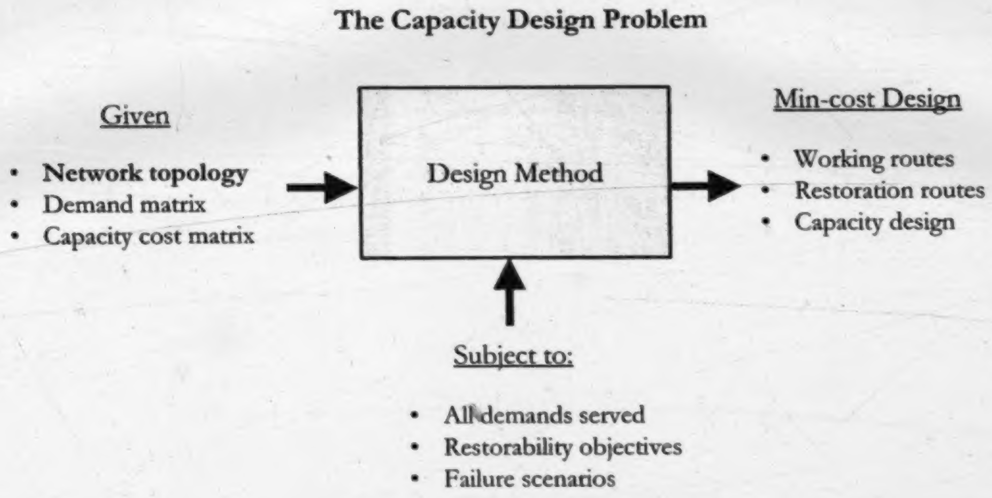

The Capacity Design Problem
Given
Network topology
Demand matrix
Capacity cost matrix
Design Method
Min-cost Design
Working routes
Restoration routes
Capacity design
Subject to:
All demands served
Restorability objectives
Failure scenarios

Figure 4 – The capacity design problem

Figure 5 shows a summary of the topology and capacity design problem. In Figure 4, we see the topology is given as an input to the problem whereas in Figure 5, the topology is an output from the design process.

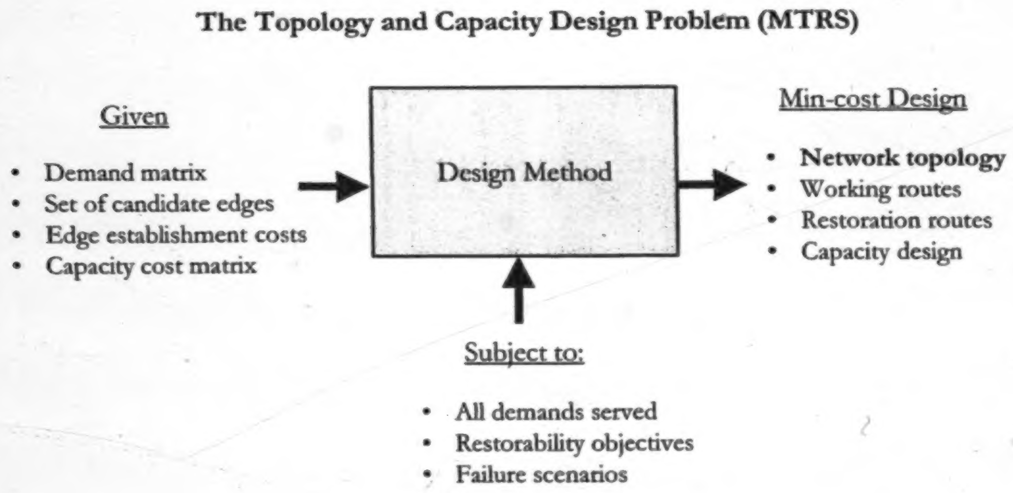

The Topology and Capacity Design Problem (MTRS)
Given
Demand matrix
Set of candidate edges
Edge establishment costs
Capacity cost matrix
Design Method
Min-cost Design
Network topology
Working routes
Restoration routes
Capacity design
Subject to:
All demands served
Restorability objectives
Failure scenarios

Figure 5 – The topology and capacity design problem

2.3 THE EFFECT OF TOPOLOGY ON NETWORK COSTS

Traditionally, in the optimal capacity (only) design problem, the facilities graph is already given. Capacity design is then carried out for this physical layout to provision capacity required for serving the working demands and achieving set restorability objectives. If we change the graph, would this change or affect the capacity design solution?

The answer is – yes. It is to be expected. Addition or removal of an edge can cause a change in the routing of demands and subsequently, in the restoration of the demands in the network. This would result in a decrease or an increase in the capacity requirements for the network and hence, the network costs. The question we try to answer in this section is the role the topology plays in the network design problem.

The capacity cost for serving a given demand matrix varies with the physical topology. The reason relates to the interdependence of the network design problem introduced in Chapter 2.1 and is developed further here. The physical topology of the network determines the paths that may be chosen for serving working demands. This directly affects the length of such paths and determines how much capacity is required on each span for routing demands. The working paths used affect/determine the route diversity available for use in restoration. This in turn determines how much spare capacity is assigned on each span. All these factors invariably impact the total cost of provisioning capacity in the network and hence, the overall network cost.

The role of network topology as either an input or output is further illustrated in Figure 6 and Figure 7.

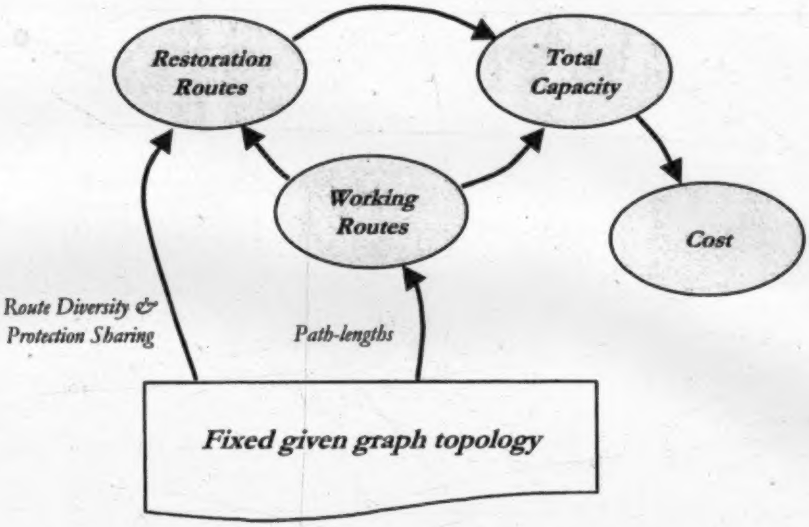

Restoration Routes
Total Capacity
Working Routes
Cost
Route Diversity & Protection Sharing
Path-lengths
Fixed given graph topology

Figure 6 – Fixed topology design problem

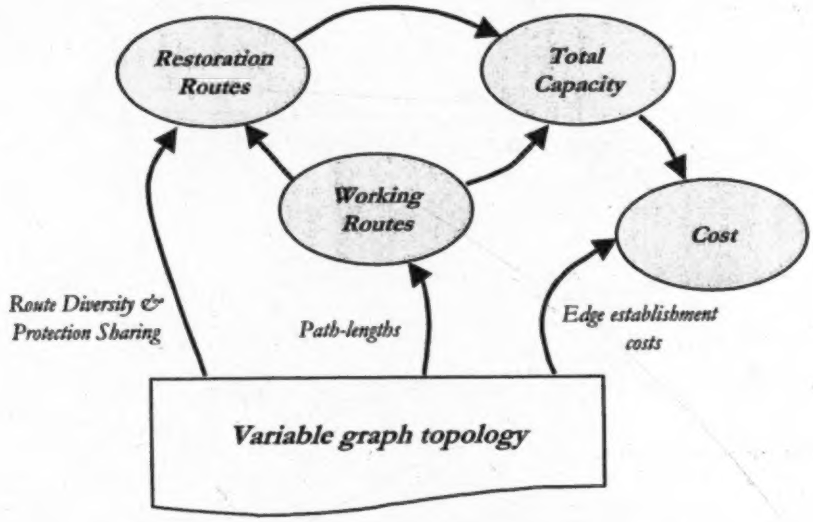

Restoration Routes
Total Capacity
Working Routes
Cost
Route Diversity & Protection Sharing
Path-lengths
Edge establishment costs
Variable graph topology

Figure 7 – Variable topology design problem

Figure 8 shows a plot of network costs against graph connectivity ($\bar{d}$). The x-axis shows increasing network connectivity in the direction from left to right, while the y-axis shows increasing network costs in the direction from down to up. This is the actual data obtained from varying the connectivity of the COST239-11n network (network data given in Appendix A) from 2.18 through 6, producing 20 different networks. The working and spare capacity for the network is obtained through a joint optimization of working and restoration route flows. There is no explicit hop-limit for each route because the problem is formulated using the node-arc integer programming (IP) model discussed in Chapter 2.4. The restoration mechanism is span-restoration. The total network cost for each network is given as the sum of the edge and capacity costs in the network.

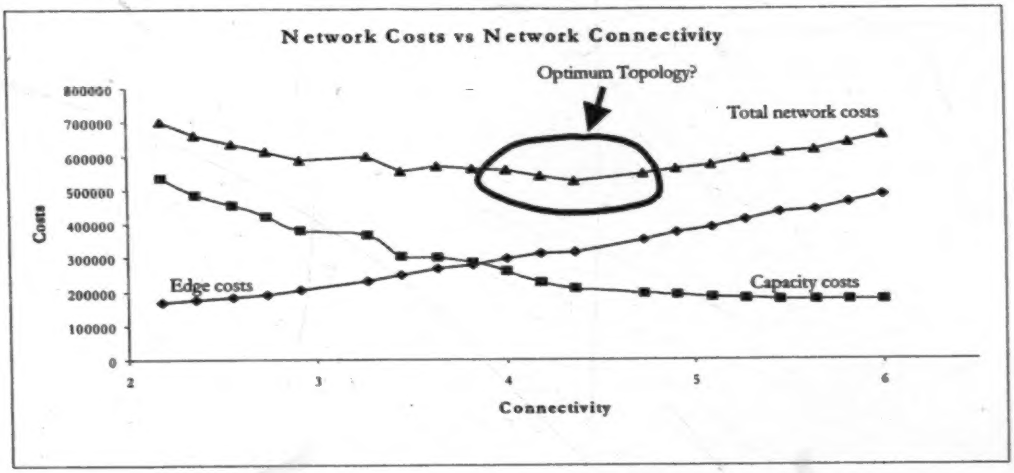


Figure 8 - Networks Costs vs. Connectivity

Going from left to right in the figure, we notice that the capacity cost of the network initially decreases as the connectivity of the network increases. This is because a more connected network topology gives rise to working paths which are shorter in length and thus require less capacity to route all demands. The network topology also offers more route diversity for the restoration/protection path and allows for greater sharing of spare capacity (capacity re-use) among multiple failure scenarios that are non-concurrent. This leads to the network requiring less spare capacity for restoration/protection and decreased capacity redundancy. Redundancy is the ratio of spare to working capacity $\left(\sum s_{total} / \sum w_{total}\right)$ - in the network. In addition, the capacity efficiency of the span-restorable mesh-restorable network topology increases with higher average nodal degrees. Generally, there will be a reduction in spare capacity by a factor of $1/\left(\bar{d}-1\right)$ as $\bar{d}$ increases [GrDo01]. Thus, $1/\left(\bar{d}-1\right)$ forms a lower bound on the redundancy required for survivability of the network.

All these factors contribute to lower the cost of provisioning capacity in the network and supports the choice of mesh transport networks as a cost effective network design approach (as opposed to traditional multi-ring networks). By itself, the decrease in capacity costs with increase in connectivity would seem to suggest a solution to the minimal capacity cost network problem that lies in the region of highly connected networks.

However, as the network connectivity increases further, we are ascending a curve for the cost of establishing each span (the fixed cost). This covers the real estate costs (acquiring the right of way for the land) and civil engineering costs (trenching, buildings, etc). These costs can sometimes be the highest single cost in establishing a network. The total cost of the network at any point is the sum of the capacity costs and the fixed cost of all spans in the network. Figure 8 shows us that there is a region between the densely and sparsely connected solutions within which the optimal topology would lie, where a balance between both costs can be achieved. The rather broad region of the optimum in this example will not always be the case. If either edge or capacity costs are dominant, a sharper turning point can be expected.

Additionally, small differences on this scale can still be hundreds of million of dollars. Therefore, the physical graph topology is introduced in the optimization problem as a variable to find the optimal topology design.

2.4 IMPORTANCE OF EDGE (SPAN) COSTS

In this thesis work, we have taken the position that it is desirable not to assume any specific data on the costs of establishing spans (the fixed span cost) and the costs of provisioning units of capacity where dark fiber exists (the unit capacity cost). This would enable our study to be general and applicable over a wider range of cost values. Therefore we represent the relationship between these two basic costs as a cost ratio Ω. At one extreme, we have a situation where the fixed costs of edge establishment dominate the capacity costs. In these cases, the minimum cost solution would tend to be based on a ring topology because this topology uses the fewest edges. At the other extreme, we have dominant capacity costs. In this case, the network solution would tend to be densely connected so as to reduce the capacity required for routing and increase the efficiency of spare capacity sharing since capacity is so expensive. In the general case of interest, the topology of choice lies between these extremes.

We define the network cost for a given set of network demands as depending on two factors:

a. The cost associated with establishing each span, F_{ij} and

b. The costs associated with adding a unit of capacity on each edge, c_{ij}

as shown in Equation (2.1):

$$C=\sum_{i=1}^{k}\Omega\cdot l_i+\sum_{i=1}^{k}(w_i+s_i)\cdot l_i \qquad \Omega=F_{ij}/c_{ij} \tag{2.1}$$

where i is an index over k spans in the network and l_i is the length of span i.

The first component $\sum_{i=1}^{k}\Omega\cdot l_i$ represents what we call the "fixed cost" of the edges employed in the solution and is affected by the value of Ω (these are the total topology costs) while the second component $\sum_{i=1}^{k}(w_i+s_i)\cdot l_i$ represents the capacity costs and depends on the total working and spare capacity (w_i+s_i) placed on each span in the network which in turn depends on the routing and restoration paths employed. It is obvious that $(w_i+s_i)=f(D)$ where D is the demand intensity matrix which the network is expected to serve because the demand intensity dictates the amount of working capacity that is placed in the network to serve the demands. This in turn dictates the amount of spare capacity that is provisioned in the network for restoration.

Equation (2.1) is equivalent to

$$C = \sum_{i=1}^{k} [\Omega + (w_i + s_i)] \cdot l_i \tag{2.2}$$

We can then say that the optimal physical topology is a function of the demand matrix, the cost of establishing each link between node pairs and the cost of provisioning a unit of capacity on each link.

2.5 PRIOR OPTIMIZATION PROBLEMS

In this section, we establish several prior optimization models that are necessary for our work and present some of the mathematical techniques and models that are usually used for solving network design problems.

Mathematical programming is used in Operations Research (OR) as the mathematical model to optimize an objective function. The syntax and structure of mathematical programming techniques provides a precise and compact language for stating, defining, and understanding the nature of problems of interest.

There are three main types of mathematical programming problems in OR; they are Linear Programming (LP), Mixed Integer Programming (MIP) and Integer Programming (IP). In LP, the objective function and all constraints must be linear functions. In addition, all decision variables must be real valued and non-negative. These variables can be continuous, i.e., they can take on fractional values. In MIP, some of the variables might be integer while others might be continuous. In IP, all decision variables are restricted to integer. It is often necessary to solve a problem as IP since fractional solutions might be meaningless in real world situations.

For example, the LP solution with the value of the decision variable for adding an edge to the network topology as 0.4 has no meaning. We can either add the edge or not. We cannot simply round the values up or down because rounding up the solution may unnecessarily increase the cost of the network. On the other hand, rounding down may jeopardize the routing feasibility or restorability of the network. In problems of this nature, the decision variables are constrained to take binary values {1, 0} to represent the decision. These problems belong to a class of IP known as "1/0 MIP". The objective function value from rounding LP solutions to integer values is often very different from the objective function value of the true IP problem solution. Therefore, it is necessary to solve certain types of problems as IP problems to enable us get feasible solutions.

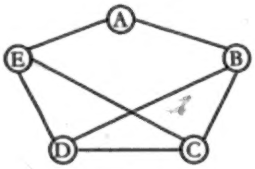
Through the use of these mathematical programming techniques in network design, we are able to obtain optimal or near-optimal solutions to many large optimization problems in reasonable amounts of time. Where this is not possible, the mathematical programming techniques provide bounds against which to measure the effectiveness of heuristic solutions that have been developed.

2.5.1 TRANSHIPMENT OR FLOW ASSIGNEMENT: A BASIC CHOICE

There are three common types of mathematical programming models: *transportation*, *assignment* and *transshipment*. The differences between the models arise from the way a problem is cast and the constraints associated with it. In this work, we use the assignment and/or transshipment models. In this section, we discuss these models and illustrate how we would represent the JCP problem using span restoration in both models for the 5-node, 7-

span (5n7s) network shown Figure 9. The purpose of developing this problem example with both models is to illustrate the difference in approach for both models and as the results show, possible differences in solutions obtained from both models.

For the network 5n7s, the spans are assumed to be of equal lengths. The demand matrix shown is generated as a uniformly distributed pattern (every node pair exchanges a demand of value 1). The restoration mechanism is span restoration.



	A	B	C	D	E
A	0	1	1	1	1
B		0	1	1	1
C			0	1	1
D				0	1
E					0

Figure 9 – Model mesh-restorable network topology (Network 5n7s)

The problem can be stated as follows:

> Minimize the amount of capacity required to route all demands exchanged between O-D pairs and to ensure survivability in the event of a single span failure. The restoration mechanism is span-restoration.

2.5.1.1 The assignment optimization model ("arc-path" formulation)

Here, the question is how to divide the total demand flow for each O-D pair over the set of different routes in the network such that the amount of capacity required is minimum. The assignment model thus deals with flow assignment to each path (made up of one or more arcs) for both routing and restoration. It is called the 'arc-path' formulation because the flow variables are assigned to paths through the network and the flow assigned to each path determines the amount of capacity (working or spare) placed on each arc of the path.

The problem seeks to:

1. assign flow values to each of the working paths such that a minimum amount of capacity is required to serve all demands and
2. assign flow values to each of the restoration paths that would be used in the event of each particular span failure such that a minimum amount of capacity is required to restore all demands affected by the failure.

We define the following notations to represent the problem in the model:

S is the set of spans in the network

D is the set of all demands.

Q^r is the set of eligible routes for routing demand r indexed by q.

$g^{r,q}$ is the amount of working flow assigned to the q^{th} route for routing demand r.

$\varsigma_j^{r,q}$ is an input parameter that is 1 if the q^{th} route for demand r uses span j and 0 otherwise.

P_i is the set of routes that may be used for restoration in the event of failure of span i indexed by p.

f_{ip} is the amount of restoration flow assigned to the p^{th} route using span i.

δ_{ij}^p is an input parameter that is 1 if the p^{th} route for restoration of span i uses span j and 0 otherwise.

w_i is the number of working capacity units added to each span i.

s_i is the number of spare capacity units added to each span i.

The formulation for the JCP problem is:

$$\textbf{JCP – Arc Path:} \qquad \min \sum_{i \in S} (w_i + s_i)$$

Subject to:

$$\sum_{q \in Q^r} g^{r,q} = d^r \qquad \forall r \in D \qquad (2.3)$$

$$w_j - \sum_{r \in D} \sum_{q \in Q^r} \varsigma_j^{r,q} \cdot g^{r,q} = 0 \qquad \forall j \in S \qquad (2.4)$$

$$\sum_{p \in P_i} f_{ip} = w_i \qquad \forall i \in S \tag{2.5}$$

$$s_j - \sum_{p \in P_i} \delta_{ij}^{p} \cdot f_{ip} \geq 0 \qquad \forall i, j \in S, i \neq j \tag{2.6}$$

$$f_{ip}, g^{r,q} \geq 0 \qquad \forall i \in S, \forall p \in P_i, \forall q \in Q^r \tag{2.7}$$

$$w_{ij}, s_{ij} = \text{integer} \qquad \forall ij \in A \tag{2.8}$$

(2.3) ensures that the total amount of flow units over all the possible paths between each O-D demand pair to be equal to the amount of demand units exchanged between the node pair. (2.4) asserts that the amount of working capacity units assigned to each span j is sufficient for all flows that use that span. (2.5) ensures that the total amount of flow over all restoration paths used in the failure of span i is equal to the amount of flow affected by the failure (the working capacity of span i). (2.6) requires the amount of spare capacity of span j to be at least equal to the largest sum of simultaneous flows over span j arising from the failure of span i. (2.7) asserts all flow assignments to be positive and (2.8) assures the integrality of the capacity on each span.

For the SCP problem, we are given the w_i values for each span. Therefore, (2.3) and (2.4) would not be included in the formulation. The SCP – Arc Path formulation is:

SCP – Arc Path: $\min \sum_{i \in S} (w_i + s_i)$

$$\sum_{p \in P_i} f_{ip} = w_i \qquad \forall i \in S \tag{2.9}$$

$$s_j - \sum_{p \in P_i} \delta_{ij}^{p} \cdot f_{ip} \geq 0 \qquad \forall i, j \in S, i \neq j \tag{2.10}$$

$$f_{ip} \geq 0 \qquad \forall i \in S, \forall p \in P_i \tag{2.11}$$

$$s_{ij} = \text{integer} \qquad \forall ij \in A \tag{2.12}$$

Using our network example, we establish the parameters required for performing JCP optimization on the given network topology. Suppose we want a route set of size two for both routing and restoration routes. We identify the two possible working routes by depth-first-search for each O-D pair in the network. Each distinct route identified is an eligible route for routing the demand associated with the O-D pair. Twenty-four eligible working routes found are shown in **Error! Not a valid bookmark self-reference.**.

Table 1 - Eligible working routes for network 5n7s

O-D pairs	Working routes	
A-B	W1 W2 W3	A-B A-E-C-B A-E-D-B
A-C	W4 W5	A-B-C A-E-C
A-D	W6 W7	A-B-D A-E-D
A-E	W8 W9 W10	A-E A-B-C-E A-B-D-E
B-C	W11 W12	B-C B-D-C
B-D	W13 W14	B-D B-C-D
B-E	W15 W16 W17	B-C-E B-A-E B-D-E
C-D	W18 W19 W20	C-D C-B-D C-E-D
C-E	W21 W22	C-E C-D-E
D-E	W23 W24	D-E D-C-E

The two shortest routes for each demand were chosen as the eligible routes (since the spans are of equal lengths, a hop-count metric has the same effect as a path-length metric). In cases where the next longer route(s) is of equal length as the 2nd shortest route, then all other routes of that length are also used. For example, the second shortest path found for demand pair A-B is of length 3 therefore all paths of length 2 are included in the set of eligible working routes. For each span in the network constituting a failure scenario, we identify two possible restoration routes also generated by depth-first-search. If the next longer route(s) is of equal length as the 2nd shortest route, then all other routes of that length are also used. Eighteen restoration routes found are:

Table 2 – Eligible restoration routes for network 5n7s

Failed span	Restoration routes	
A-B	R1	A-E-C-B
	R2	A-E-D-B
A-E	R3	A-B-C-E
	R4	A-B-D-E
B-C	R5	B-D-C
	R6	B-A-E-C
	R7	B-D-E-C
B-D	R8	B-C-D
	R9	B-A-E-D
	R10	B-C-E-D
C-D	R11	C-B-D
	R12	C-E-D
C-E	R13	C-D-E
	R14	C-B-A-E
	R15	C-B-D-E
D-E	R16	D-C-E
	R17	D-B-A-E
	R18	D-B-C-E

Taking the r^{th} O-D pair to be A-B with working routes W1 and W2 and $d = 1$, then the assignment constraint is:

$$W1 + W2 = 1 \tag{2.13}$$

Thus, the set of constraints for the routing of demands is given as follows:

W1 + W2 + W3 = 1
W4 + W5 = 1
W6 + W7 = 1
W8 + W9 + W10 = 1
W11 + W12 = 1
W13 + W14 = 1
W15 + W16 + W17 = 1
W18 + W19 + W20 = 1
W21 + W22 = 1
W23 + W24 = 1

If the amount of working flow carried by span *i* prior to its failure is w_i, then the restoration constraint contributed by span *i* is given as:

$$R1 + R2 = w_{A-B} \tag{2.14}$$

where span *i* is span A-B. In a similar fashion, the set of constraints relating to the restoration of demands impacted by a span failure is obtained as follows:

$R1 + R2 = w_{A-B}$
$R3 + R4 = w_{A-E}$
$R5 + R6 + R7 = w_{B-C}$
$R8 + R9 + R10 = w_{B-D}$
$R11 + R12 = w_{C-D}$
$R13 + R14 + R15 = w_{C-E}$
$R16 + R17 + R18 = w_{D-E}$

From the solution to the flow assignment problem, we have the following capacity valued added to each span:

SPAN	WORK	SPARE
A-B	2	2
A-E	2	2
B-C	2	1
B-D	2	1
C-D	1	1
C-E	2	1
D-E	2	1
Total	13	9

We now present the same problem using the transshipment model.

2.5.1.2 The transshipment optimization model ("node-arc" formulation)

The transshipment problem is one that allows the shipment of goods (in this case, demand units) between the supply and demand points and may contain transshipments points through which the goods may be shipped on their way from a supply to a demand point. For an O-D pair, the supply point is the origin node and the demand point is the destination node. The demand between the origin and destination nodes can be transported through other nodes in the network (the transshipment nodes). The transshipment model is also called the 'node-arc' formulation because the flow variables being solved are associated with the network arcs (edges) from a node (the source, sink or transshipment) and not with any paths per se (as used in the arc-path formulation).

The origin node can neither be a sink nor a transshipment node for the demand. Similarly, the destination node cannot be a source nor transshipment node for the demand. The transshipment nodes can both receive and send out the demand flow but cannot sink the flow. Thus, in the transshipment problem formulation, we have three major sets of constraints governing the behavior of the source, sink and transshipment nodes. Equations (2.15) to (2.17) show the constraints for routing the r^{th} O-D demand pair with value of d_r between its origin node *i* and its destination node *j* through any transshipment node *k*.

Equation (2.15) is the constraint for the origin node: the sum of all outflows from origin node *i* to its adjacent neighbors *k* is equal to d_r for routing the r^{th} demand quantity.

$$\sum_{k=1}^{m} x_{ik} = d_r \tag{2.15}$$

Equation (2.16) is the constraint for the destination node: the sum of all inflows to destination node *j* from its adjacent neighbors *k* is equal to d_r for the r^{th} demand quantity.

$$\sum_{k=1}^{m} x_{kj} = d_r \qquad (2.16)$$

Equation (2.17) is the constraint for the transshipment nodes: the sum of all inflows to the transshipment node *k* from its adjacent neighbor nodes *i* is equal to the sum of outflows to its neighbor nodes *j* for the r^{th} demand quantity.

$$\sum_{i=1}^{m} x_{ik} - \sum_{j=1}^{m} x_{kj} = 0 \qquad (2.17)$$

The total sum of flows over all paths for demand pair *r* is then equal to d_r. For restoration, the constraints are obtained in a similar manner with the end nodes of the failed span replacing the origin and destination nodes of the O-D pair.

The complete formulation for the transshipment problem is:

JCP – Node Arc: $\sum_{ij\in S}\left(w_{ij}+s_{ij}\right)$

Subject to:

$$\sum_{nj\in S} w^r_{nj} = d^r \qquad \forall r\in D, n=O[r] \qquad (2.18)$$

$$\sum_{jn\in S} w^r_{jn} = d^r \qquad \forall r\in D, n=T[r] \qquad (2.19)$$

$$\sum_{in\in S} w^r_{in} - \sum_{nj\in S} w^r_{nj} = 0 \qquad \forall r\in D, n\notin \{O[r],T[r]\} \qquad (2.20)$$

$$w_{ij} = \sum_{r\in D} w^r_{ij} \qquad \forall ij\in S \qquad (2.21)$$

$$\sum_{ik\in S;\, j\neq k} s^{ik}_{ij} = w_{ij} \qquad \forall ij\in S \qquad (2.22)$$

$$\sum_{kj\in S;i\neq k} s_{ij}^{kj} = w_{ij} \qquad \forall ij\in S \qquad (2.23)$$

$$\sum_{nk\in S;k\notin\{i,j\}} s_{ij}^{nk} - \sum_{kn\in S;k\notin\{i,j\}} s_{ij}^{kn} = 0 \qquad \forall ij\in S, \forall n\notin\{i,j\} \qquad (2.24)$$

$$s_{kl} \geq s_{ij}^{kl}, s_{kl} \geq s_{ji}^{kl} \qquad \forall (ij),(kl)\in S^2, (ij)\neq(kl) \qquad (2.25)$$

(2.21) ensures that the sum or working capacity assigned to span ij is equal to the sum of working flow that span ij for each demand pair r. (2.22) asserts that the outflow from the origin of any failed span ij is equal to the amount of working flow carried by that span (before the failure) and (2.23) asserts that the inflow to the destination node of the failed span ij is equal to the amount of flow carried by the span. (2.24) is the constraint for transshipment nodes used in restoration similar to (2.17).

For the SCP problem, we are given the w_{ij} values for each span. For the transshipment SCP model formulation, we have:

SCP – Node Arc: $\sum_{ij\in S} s_{ij}$

Subject to:

$$\sum_{ik\in S;j\neq k} s_{ij}^{ik} = w_{ij} \qquad \forall ij\in S \qquad (2.26)$$

$$\sum_{kj\in S;i\neq k} s_{ij}^{kj} = w_{ij} \qquad \forall ij\in S \qquad (2.27)$$

$$\sum_{nk\in S;k\notin\{i,j\}} s_{ij}^{nk} - \sum_{kn\in S;k\notin\{i,j\}} s_{ij}^{kn} = 0 \qquad \forall ij\in S, \forall n\notin\{i,j\} \qquad (2.28)$$

$$s_{kl} \geq s_{ij}^{kl}, s_{kl} \geq s_{ji}^{kl} \qquad \forall (ij),(kl)\in S^2, (ij)\neq(kl) \qquad (2.29)$$

The MIP solution to this capacity design problem assigns the following capacity values to each span:

SPAN	WORK	SPARE
A-B	2	2
A-E	2	2
B-C	3	1
B-D	1	1
C-D	1	1
C-E	2	2
D-E	2	0
Total	13	9

For the small network test example used here, the total amount of working and spare capacity placed in the network in the solution from both models is the same though the distribution over the spans in the network is different. Where a different distribution of demands between O-D pairs is used or the spans have different costs for adding capacity on the edge, this difference in capacity placed on the edges may result in cost differences between the solutions obtained from both optimization models.

2.5.2 CHOICE OF TRANSHIPMENT MODEL FOR SOLVING MTRS PROBLEMS

In the previous section, we presented a problem using the transshipment and assignment programming models and we saw that the choice of model may lead to different solutions. In this section, we point out the major difference in both models that potentially causes the difference in solutions. We identify which model and techniques we would employ in this work.

The assignment problem formulation allows us explicit control over the routing of demands in the network since we have a choice of paths to include in the design. We can observe and manipulate the working and restoration routes used in the solution. Such control allows trade-off between the run times and solution quality, as a user is able to specify the number of routes included in the problem for routing and restoration. Note that increasing the size of the route set results in increases in the run-time and solution quality and vice versa.

In contrast, the transshipment problem formulation does not offer any control over the paths used for routing and/or restoration because the paths are not explicitly defined. The lack of explicit control removes any limitation of maximum distance or hop count over paths chosen. The implication of this is that the eligible route set contains all possible routes between each origin-node pair and may lead better solutions (lower cost) than a corresponding solution where the route set is limited by some metric. The constraint that total source flow equal total sink flows essentially forces the spans in the transportation problem to be modeled as unidirectional having specific origin and destination.

When the topology itself is a variable in the problem, defining the set of working or restoration routes may not be feasible (except in networks of small sizes) because a set of eligible routes for each possible graph topology has to be developed ahead of time. Thus, we use the transshipment design model to formulate the topology and capacity design (MTRS) problem.

For the most part in this work, we consider the problem of simultaneously optimizing both topology and capacity design for the network. In the Sweep Search heuristic (presented in Chapter 6.3), the capacity optimization method used is the span-restorable JCP design (transshipment model) so as to enable us to obtain the optimal span-restorable capacity design solution for each generated qualified topology.

Before we go any further into the subject of network topology design, we will take a look at what currently is a feasible topology for some service providers. The next section deals with this.

2.6 A LOOK AT SOME REAL NETWORKS

The network layer can be partitioned into sub-networks according to some geographic or administrative boundaries. For instance, it is common to partition the layer into separate access sub-network, metropolitan inter-office (metro) and core (long-haul) sub-networks. An access network connects end users to the backbone network. The demands originate from customer premises and remote switching offices and terminate at a main switching office (called a hub). A metro sub-network connects the main switching offices or other points of concentration within the metropolitan area. Long haul sub-networks connect metropolitan areas on a national or international scale. In this way, demands aggregate through the layers in the network. The transport networks we consider are for metro and long haul/regional networks.

In this section, we look at typical fiber route topologies for these networks. Many networks being built are based on rights-of-way obtained from a utility company (power, gas, railway) and so, inherit the topological properties of those networks. As a result of either a utilities origin or legacy evolution of an existing operators route structure, typical transport network graphs have an irregular partial interconnection pattern with an average number of geographically diverse rights-of-ways at junction sites that varies from two to at most about seven. In some North American fiber networks the average degree may be as low as 2.25 indicating a sparse topology. A degree of two at every node is the minimum possible in any network that has a fully restorable topology. In some European networks the average degree is as high as 4.5.

Figure 10 to Figure 14 shows the fiber topology layouts for some European and North American networks obtained from published data. The Genuity network is its fiber optic backbone for the United States. The Level 3 United States network provide communication between 48 cities with approximately 40,000 km route miles. JANET is the academic and research network for the United Kingdom (as of spring 2001). Figure 13 is the Belgian national transport network. Figure 14 is the pan-European COST 239 transport network as used in developed for action (project) 239, studying Ultra-High Capacity Optical Transmission Networks. COST is a European-based intergovernmental organization for European Co-operation in the Field of Scientific and Technical Research (see http://cost.cordis.lu/src/home.cfm).

We see the Level 3 and Genuity networks having low degrees of connectivity with very few nodes having a nodal degree of 3 or greater. In contrast, we can see there is a greater degree of mesh connectivity in the Belgian and COST239 networks. The Level 3 network has a nodal degree of ~2.25 (the Genuity network has a similar value) while the Belgian network has a nodal degree of ~3.02 and the COST239 network has a nodal degree of ~4.73. The JANET network shows a backbone topology having 8 nodes and 10 spans (nodal degree of 2.5) with a number of access nodes. Metropolitan area networks in both North America and Europe tend to be similar to the more richly connected European national backbone example in Figure 13 whereas long-haul networks in North America tend to be sparser.

The general architecture of these networks suggests some important characteristics of feasible transport networks. The graphs all have relatively low average nodal degrees. Even the highest average nodal degree we can realistically imagine, say $\overline{d} = 5$, is still extremely lightweight compared to a truly random or fully connected graph. The transport network graphs tend to look neither like highly connected networks nor like random graphs where any node may with equal probability be connected to any other. The graphs of real networks tend to show a high degree of locality in that nodes tend to be connected to other geographically adjacent node. In other words, we do not see spans crossing the plane from one node to the other. The graphs are also not strictly planar but if non-planar, have relatively few out-of-plane spans. In developing our heuristics, we draw upon and make use of these properties.

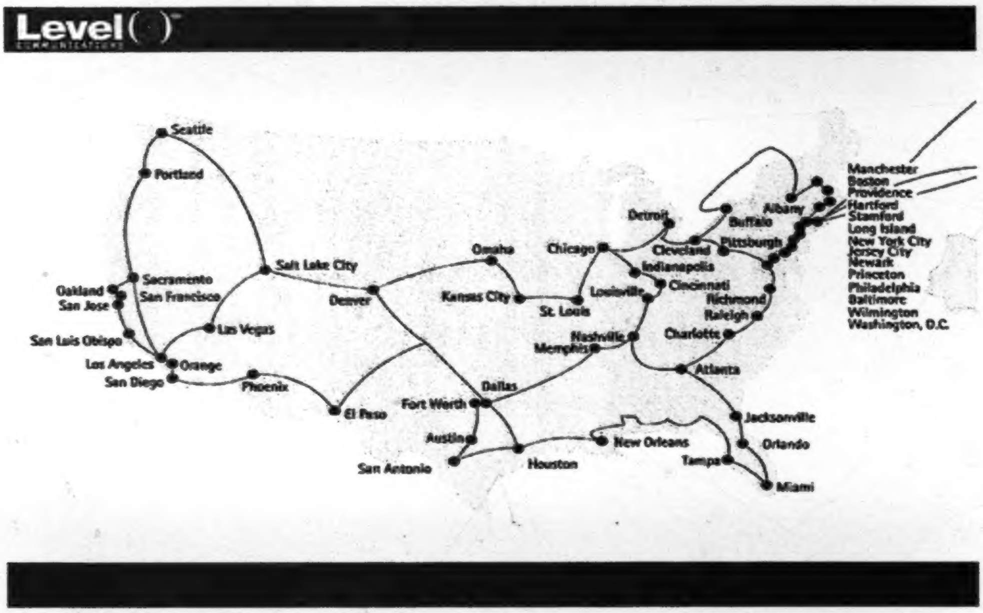

Level( )
Seattle
Portland
Sacramento
San Francisco
Oakland
San Jose
San Luis Obispo
Los Angeles
Orange
San Diego
Las Vegas
Phoenix
Salt Lake City
Denver
El Paso
Omaha
Kansas City
St. Louis
Chicago
Detroit
Indianapolis
Louisville
Cincinnati
Nashville
Memphis
Dallas
Fort Worth
Austin
San Antonio
Houston
New Orleans
Atlanta
Charlotte
Raleigh
Richmond
Cleveland
Pittsburgh
Buffalo
Albany
Jacksonville
Orlando
Tampa
Miami
Manchester
Boston
Providence
Hartford
Stamford
Long Island
New York City
Jersey City
Newark
Princeton
Philadelphia
Baltimore
Wilmington
Washington, D.C.

Figure 10 – Level 3 Fiber Network for the United States

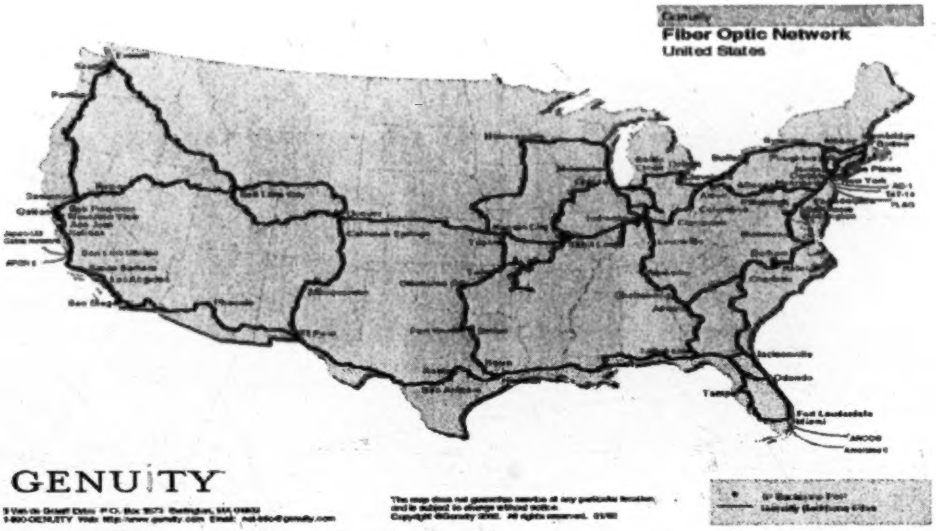

Genuity
Fiber Optic Network
United States
New York
GENUITY

Figure 11 – Genuity Fiber Optic Network for the United States

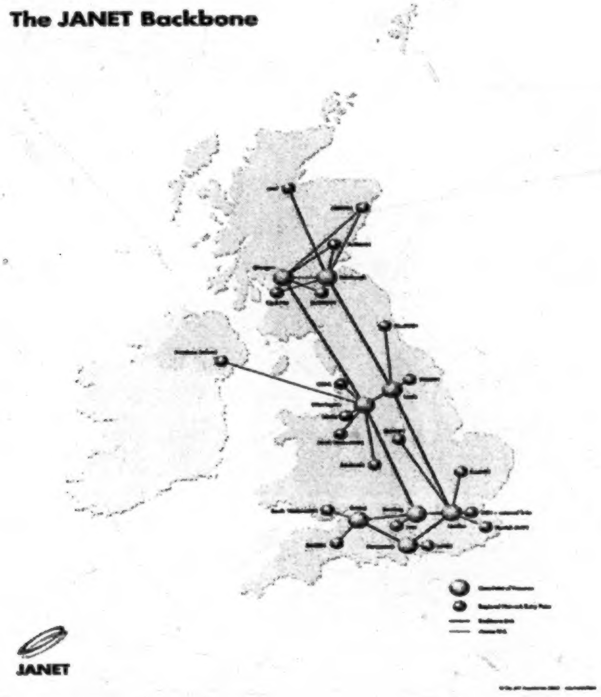

The JANET Backbone
JANET

Figure 12 – The JANET backbone network for the United Kingdom

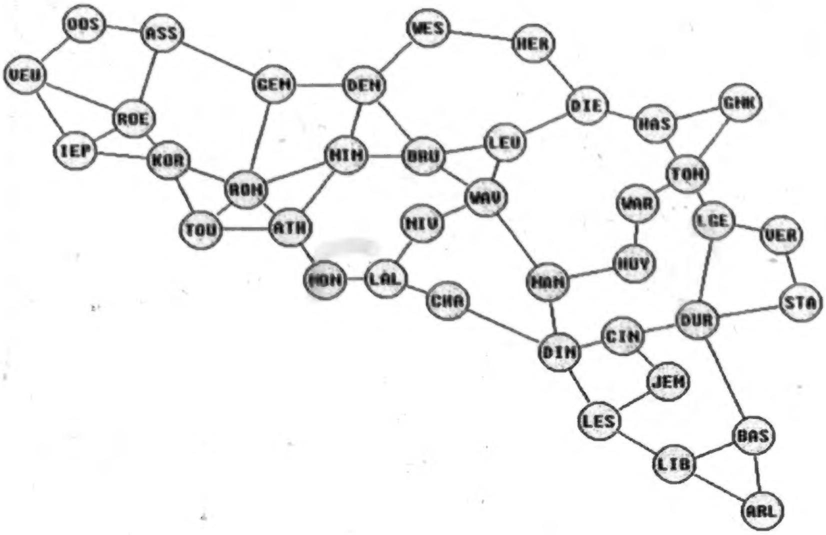

OOS
ASS
VEU
ROE
IEP
KOR
GEN
DEN
WES
HER
DIE
HAS
GNK
NIN
BRU
LEU
TON
RON
WAV
WAR
TOU
ATH
NIV
LGE
VER
MON
LAL
CHA
NAM
HUV
STA
DUR
CIN
DIN
JEM
LES
BAS
LIB
ARL

Figure 13 – The Belgian Transport network

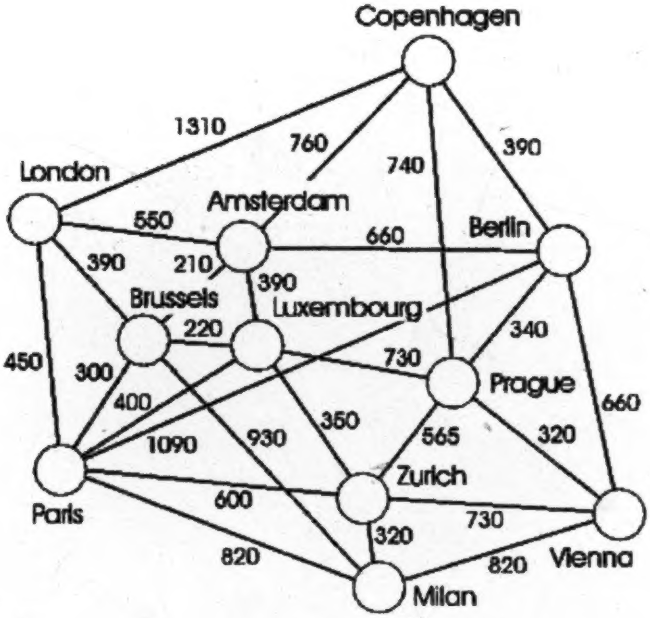

Copenhagen
1310
760
740
390
London
Amsterdam
550
660
Berlin
390
210
390
Brussels
Luxembourg
220
340
450
300
730
Prague
400
660
1090
930
350
565
320
Zurich
600
Paris
730
320
820
820
Vienna
Milan

Figure 14 – COST239 European Model Network

2.7 ELEMENTAL GRAPH THEORY AND ALGORITHMS

The physical facilities network is represented as a graph and graph theoretic principles are used in developing the heuristics proposed. In this section, we present the graph theoretic principles employed in this work and the basic algorithms that are used. These algorithms will be used in testing for the properties we deem necessary for a feasible network in any proposed graph topology. We start with the mathematical background for set theory notations that will be used.

A *set* is a collection of distinct objects (called its **elements** or **members**). When x is a member of set A, we denote this as $x \in A$ and say "x **belongs to** A". When x is not in A, we write $x \notin A$. If every element of set B belongs to set A, we say that B is a **subset** of A, and A **contains** B. We write $B \subseteq A$ or $A \supseteq B$ respectively.

We can explicitly list the elements of a set A. For example, we can write $A = \{1,0\}$. Sets A and B are **equal** when they contain the same elements and we write this as $A = B$. A **proper subset** of set A is a subset that is not A itself. However, every set is a subset of itself. The **empty set** $\varnothing$, contains no elements and is a subset of every set. We write $x, y \in S$ to show that both x and y are elements of S and $x \neq y$ to show that x and y are not same.

The number of elements in a finite set A is its **size** and it is written as $|A|$. For a graph $G = (V, E)$, V is its vertex (or node) set and E is its edge set. The number of edges corresponding to a full mesh on $v = |V|$ vertices is given by $|E| = \binom{v}{2} = v(v-1)/2$. If E is complete, we refer to it as representing the **universe** of all possible edges (the universal edge set) for a topology determination problem. $\forall$ is a universal quantifier that expresses the understanding "for every x in A" or " for all x in A" and we write this as $\forall x \in A$.

We now turn our attention to the terms used in association with graphs. In [West01], we have the definition of a graph as "a triple consisting of a vertex set $V(G)$, an edge set $E(G)$ and a relation that associates with each edge two vertices (not necessarily distinct) called its end points. In [Wils96] and [Temp81], a graph is defined as a collection of points or vertices, some pairs of which may be joined by lines or edges. In the transport network context, the set of switches are the vertices of the graph and the physical fiber cables that connect the switches together make up the edges. We write $e=(i,j)$ (or $e=(j,i)$) for an edge e with end points i and j. An edge whose end points are the same is a loop. However, we consider only loop-less networks where the end nodes of each edge are distinct.

There may exist multiple edges between any node pair in a graph. Where this exists, the graph is called a multi-graph. A **simple graph** is a graph with no loops or multiple edges. [Wils96] defines a simple graph G "as a non-empty finite set $V(G)$ of elements called vertices, and a finite set $E(G)$ of distinct unordered pairs of distinct elements of $V(G)$ called edges". We model the physical topology on which a transport network is realized as a simple graph. In reality, the transport network is a multi-graph where the physical cable between any two switches is made up of multiple fibers and each fiber can carry a number of wavelengths. However, we group all fibers between distinct node pairs together as an edge, thus giving our model of a simple graph with distinct edges.

Graphs can be *directed*, if traffic can only go in one direction on an edge (called digraphs) or *undirected*. The transport network is an undirected graph with traffic going either way on an edge.

A *path* is a sequence of contiguous edges or nodes through the graph that connect two vertices. A cycle is a path with an equal number of vertices and edges whose vertices can be placed around a circle so that two vertices are adjacent if and only if they appear consecutively along the circle [West01]. A walk is a sequence of ordered edges from one vertex to another, a path is a walk with distinct vertices and a cycle is a walk that ends on the vertex where it started.

Graph Traversal:

A graph traversal routine is a necessary part of many algorithms that solve problems on graphs. A traversal is a journey through the graph, which visits every vertex and edge. We use the graph traversal algorithm to establish that a graph is connected. This algorithm is a recursive depth-first search routine described in [BaVG00] with both space and time complexity of $\Theta(n+m)$ where $n=|N|$ and $m=|E|$. The basic concept is to "explore if possible, otherwise backtrack" where exploration is the arrival at or discovery of an unvisited node v from a node u, backtracking is the return to u from the node v.

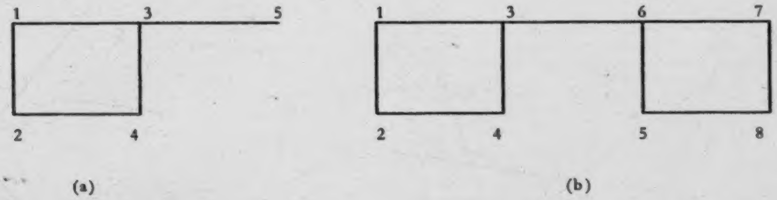
The pseudocode for the recursive depth-first search from a starting node u is:

```
Dfs (G, u):
    Mark $u$ as discovered.
    For each vertex $v$ such that edge $(u,v)$ is in $G$:
        If $v$ is undiscovered:
            dfs(G, v); that is, explore vw, visit w, explore from there as much as possible, and backtrack
                from w to v, where w is adjacent to v.
            Mark v as finished.
```

Two-edge connected graphs:

For a connected graph G, its edge connectivity $\lambda(G)$ is the size of the smallest cut-set in G. A cut-set in G is the minimum number of edges we need to delete to disconnect G. In other words, the removal of a cut-set from G increases the number of (unconnected) components in G. We can then say that G is *k-edge connected* if $\lambda(G) \geq k$ [Wilson01].

For the two-edge connected test, the only condition for which a network is not two-connected is when it contains an edge (called a bridge) that joins two components as shown in the figure below. If a component is identified by the presence of a cut node *v*, and the parent of *v* is also another cut node, then the edge *vw* where *w* is the parent of *v*, is a cut span. In the graphs shown below, the bridge in (a) is edge 3-5 and edge 3-6 is the bridge in (b). Such network topologies (having stubs and bridges) do not provide paths for restoration and are not survivable.



Two-node connected graphs:

The concepts here are analogous to those outlined above using nodes, instead of edges. If a graph G is connected and is not complete (having an edge between each node pair), its (node) connectivity $\kappa(G)$ is the size of the smallest separating node set in G. A separating set in G is a set of nodes whose removal disconnects G into bi-components where a bi-component is defined as a connected sub-graph of G, i.e. a bi-connected sub-graph not contained in any larger bi-connected sub-graph [BaVG00]. If the separating set contains only one node, the node is called a cut vertex (or bridge node) [Wils96]. We then say that G is *k-connected* if $\kappa(G) \geq k$.

A bi-connected transport network has at least two node-disjoint paths between any node pair [Suur97]. The implication of this requirement for our purposes is that any feasible graph must be at least 2-connected (or bi-connected). To this effect, we require graphs having $\kappa(G) \geq 2$. This means that at least two nodes in G must be removed before the graph is disconnected. Simply defined, a bi-connected graph is one that is connected and contains no cut vertex set. The two-edge connected network can provide restoration paths in failure scenarios involving node loss. However, only a two-node network can provide restoration paths when the failure is from a node loss.

Two-(edge or node) connected test algorithm:

The test for two-edge (or node) connectivity is perhaps one of the most important tests we will use to determine a feasible transport network graph. Therefore the algorithm used is run very frequently in assessing the feasibility or merit of any proposed graph. It is used in Tests 5 and 6 of the Graph Sifter (Chapter 5) to test graphs proposed by the graph generator. In the Sweep Search heuristic, the pare-down routine removes edges from the graph subject to survivability constraints (the graph remains two-node connected). This algorithm is used to check the validity of each edge removal. If removal is invalid (does not satisfy the survivability constraint), the edge is not removed.

Let us now describe the algorithm used, which is taken from [BaVG00]. The definition of some terms follows first.

Root node: The starting vertex for the tree

Ancestor: v is an ancestor of vertex w in a tree if v is on the path from the root to w.

Proper ancestor: Vertex *v* is a proper ancestor of vertex *w* if *v* is an ancestor of *w* and $v \neq w$.

Parent: The closest proper ancestor of a vertex *v* is the parent of *v*.

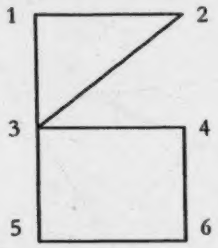
Back edge: If vertex *w* is an ancestor of vertex *v*, then edge *vw* is called a back edge.

The tests for two-edge and two-node connectivity use the same principle with the difference being that the two-node connectivity test checks for a cut node, while the two-edge connectivity test checks for a cut span that disconnects the graph. In a depth-first search tree, a vertex *v* other than the root is an articulation node (cut node) if and only if *v* is not a leaf and some sub-tree of *v* has no back edge incident with a proper ancestor of *v* [BaVG00].

The bi-component algorithm tests to see if a vertex in the depth-first search tree is an articulation point (cut-node) each time the search backtracks to the node. From a root vertex in the graph, a depth-first search tree is constructed. From a root vertex *v*, all its incident edges are explored. To explore an edge means to traverse it if it has not been traversed. For each edge *vw* (*w* is an adjacent vertex to *v*) not traversed, *w* becomes the root for a sub tree. With each node visited, its depth in the tree is recorded. Unless a back edge is encountered, the back value is taken to be its depth in the tree. The back value is the lowest depth from the root node to which a node can reach.

When a sub tree rooted at a node has been exhausted (all adjacent neighbors have been reached), a process of back tracking is begun from the last node encountered. If a node *w* is incident to a proper ancestor of its parent *v*, then there exists a back edge for *w*, whose depth is less than that of *v*. Suppose the proper ancestor is vertex *u*. Thus the back depth for *w* and *v* is updated to reflect the depth of *u* in the tree. If the least depth *w* can reach back to is that of *v*, then *v* is an articulation point for all edges in the sub tree rooted at *v*, as long as *v* is not the root vertex for the tree. This process continues till the starting vertex is reached.

This algorithm is illustrated using the network shown below.



We need arrays for storing:

1. the depth in the tree for each vertex,
2. the parent of each vertex and
3. how far up in the tree each node can reach (back value).

Each array is of size *n* where *n* is the number of vertices in the graph. We initialize the values of all depths and back values to 0. A zero value indicates that the node has not been visited.

Suppose vertex 1 is our root node. We assign its depth in the tree as 1 and as the root node, it has a back value of 1. Thus, we have that depth[1] = 1, back[1] = 1. Its incident edges are (1,2) and (1,3) and neither has been traversed. For each of the edges, we visit the incident nodes. Suppose we start from (1,2). For node 2, we assign depth[2] = 2, back[2] = 2. The incident edges are (1,2) and (2,3). The edge (1, 2) has been traversed. The edge (2,3) is then chosen. Node 3 gets depth[3] = 3, back[3] = 3.

The incident edges to node 3 are (1,3), (2, 3), (3, 5) and (3, 4). The edge (1, 3) has not been traversed and because it leads to a proper ancestor of 2 (the parent of 3), it is a back edge for node 3. The back value for 3 is adjusted to back[3] = 1, where 1 is the minimum value between the depth of node 1 in the tree and the current back value of node 3 which is 3. The edge (2, 3) has been traversed so it is not returned to. Suppose edge (3, 5) is traversed next.

Node 5 gets depth[5] = 4, back[5] = 4. Incident edges are (3, 5) and (5, 6). So edge (5, 6) is traversed. Node 6 gets depth[6] = 5, back[6] = 5.

Incident edges are (5, 6) and (4, 6). Node 4 is then visited by traversing edge (4, 6). At node 4, depth[4] = 6, back[4] = 6. The incident edges are (3, 4) and (4, 6). Since (edge (4, 6) has been traversed, edge (3, 4) is chosen. Since node 3 is a proper ancestor of node 6, back[4] is updated to be the minimum value between the current back[4] and depth[3]. So back[4] = 3.

Node 4 is the last node encountered so we start back tracking from node 4. We backtrack from node 4 to node 6 and compare back[4] with depth[6]. If back[4] ≥ depth[6], this means that node 4 has no other connection with any other node except through node 6 and therefore, node 6 is a cut node. If however, back[4] < depth[6], it means node 4 has a connection with another node which is a parent of node 6. In this case, back[4] = 3 and depth[6] = 5, so node 6 is not a cut node. Node 6 then gets a back value equal to the minimum value between back[4] and back[6]. This is because if a descendant of a node can reach a node, then the ancestor can reach the same node through its descendant. So back[6] gets 3.

From node 6, we backtrack to node 5. Again, we compare the back[6] and depth[5] values. back[6] = 3, depth[5] = 4. What this means is that node 6 can reach higher in the tree than node 5. Node 5 then gets a back value equal to min (back[5], back[6]). back[5] = 3.
We backtrack from node 5 to node 3. Comparing back[5] with depth[3], we find that the values are equal. This means that node 3 is a cut node for the sub-tree rooted at node 3 itself. A component (consisting of nodes 3-4-5-6) is thus identified. The algorithm continues in a similar manner until the root node for the tree is reached and all cut nodes are identified.

The pseudocode is given as:

```
Stack edgeStack; discoverTime = 0;
Components (G, u):
    Mark $u$ as discovered.
    discoverTime++; depth[u] = discoverTime; back[u] = discoverTime;
    For each vertex $v$ such that edge $(u,v)$ is in $G$ :
        If $v$ is undiscovered:
            parent[v] = u;
            Push edge $(u,v)$ onto edgeStack;                                  (tree edge)
            Components (G, v); that is, explore vw, visit w, explore from there as much as possible, and
            backtrack from w to v, where w is adjacent to v.
            If (back[v] ≥ depth[v])                                     (does u cut off v? - yes)
                Pop edges from edgeStack until edge $(u,v)$ is popped. bi-component found rooted at v.
            Else                                                         (does u cut off v? - no)
                back[v] = min(back[v], back[w])
        Else if (depth[v] < depth[u] and v != parent[u])                          (back edge)
            Push edge $(u,v)$ onto edgeStack; back[u] = min(back[v], back[w])
```

Analysis: This is an undirected depth-first search with appropriate processing of edges and nodes encountered. Each node is visited once and from there, recursive calls are made to each adjacent node to perform the depth search tree algorithm. Each edge is traversed only once. Therefore, the algorithm has both space and time linear complexity of $\Theta(n+m)$ where n = number of nodes and m = number of edges, as given in [BaVG00].

Minimum Spanning Trees:

Next, we introduce the Minimum Spanning Tree algorithm that we use in the graph generator described later in Chapter 4. A spanning tree for a connected, undirected graph G

is an undirected tree that contains all the vertices of G, on the condition that G is connected. A minimum spanning tree (MST) for a weighted graph is then a spanning tree with minimum weight or cost [BaVG00]. For any graph $G = (V, E)$ with $v = |V|$, there are v^{v-2} distinct labeled trees on v vertices [Wils96]. There may be multiple distinct spanning trees for a given graph. We use Prim's algorithm described in [BaVG00] to obtain a MST for a graph. This implementation has a worst-case time complexity of $\Theta(n^2)$ and a lower bound of $\Theta(m)$ and a space complexity of $3n$.

The pseudocode for the algorithm is:

```
PrimMST (G):
    Initialize all vertices as unseen vertices.
    Select an arbitrary vertex s to start the tree. Reclassify s as a tree vertex.
    Reclassify all vertices adjacent to s as fringe vertices.
    While there are fringe vertices:
        Select an edge of minimum weight between a tree vertex t and a fringe vertex v
        Reclassify v as tree vertex; add edge $(t, v)$ to the tree
        Reclassify all unseen vertices adjacent to v as fringe vertices.
```

In [MoMP90], the authors show that there exists an optimal two-connected graph whose vertices all have degree 2 or 3, such that the removal of any edge or pair of edges leaves a bridge in the component, if the distance functions of the graph satisfy the triangle inequality i.e.

a. $d(u,u) = 0$

b. $d(u,v) \geq 0$

c. $d(u,v) = d(v,u)$

d. $d(u,w) \leq d(u,v) + d(v,w)$

$\forall u, v, w \in V$, where $d(u,v)$ is the weight or length of the edge (u,v). We make use of this conclusion in developing the heuristic presented in this work.

3. MESH TOPOLOGY, ROUTING AND SPARING (MTRS)

In this chapter, we present the formal statement of the mesh-based transport network design problem *including determination of the optimum graph topology*. We show the MIP structure of the complete problem we are trying to address. The solution to the MIP problem supplies a reference solution against which we evaluate our heuristics. The entire problem definition and statement is abstracted from [GrDo01].

3.1 INTRODUCTION

The mesh topology, routing and spare capacity (MTRS) problem considers the joint optimization of the network topology, the working routes and restoration capacity. The complete problem simultaneously involves:

1. The selection of a set of edges comprising a closed connected graph,
2. Routing and provisioning of capacity for working flows,
3. Provisioning of restoration routes and spare capacity, such that all demands are served and the network is fully span-restorable against any single edge failure,

at a minimum cost.

The network cost we model is made up of two components: A one time cost, the fixed cost, which represents the cost for acquisition and preparation of the facilities route (includes costs of legal right of way contracts, digging a ditch between each node pair, the installing conduits in the ditch, the cable system itself, etc.), and an incremental cost, the unit capacity cost, for each individual wave-length or wave band turned on each span (includes the regenerator costs and line terminating equipment) [PoDe97].

For its formal modeling as a MIP optimization problem, the MTRS problem is represented using the node-arc flow (transshipment) structure because the topology is included as a variable in the problem formulation and therefore, we cannot make explicit representations of possible paths for working or restoration routes.

For this and subsequent formulations, we use the following notation:

Parameters (inputs):

N is the number of nodes, N is the set of all nodes

E is the set of edges in the graph on N nodes

D is the set of all non-zero demand quantities exchanged between nodes, indexed by r

d^r is the amount of demand associated with the r^{th} demand pair in D

$O[r]$ is the origin node for the r^{th} demand pair in D

$T[r]$ is the target or destination for the r^{th} demand pair in D

$c_{ij}(=c_{ji})$ is the cost of adding a unit of capacity to edge (i, j) (if edge (i, j) exists)

F_{ij} is the fixed cost for establishment of an edge (i, j) in the graph.

K is an arbitrary large positive number, larger than any expected accumulation of working capacity on any one edge in the solution.

Variables (outputs):

w_{ij}^r is the amount of working flow routed over the edge (i, j) for relation r

w_{ij} is the working capacity assigned to the edge (i, j) to support all working flows routed over that edge

s_{ij}^{ik} is the amount of restoration flow routed over the edge (i,k) for the restoration of failed edge (i, j)

s_{ij} is the spare capacity assigned to the edge (i, j) to support the largest combination of simultaneously imposed restoration flow requirements

$\delta_{ij} = \delta_{ji}$ is the 1/0 decision variable indicating whether the edge (i, j) exists in the graph or not (equals 1 if edge is selected, 0 otherwise).

The MTRS MIP problem can then be stated as:

MTRS:

$$\text{Minimize} \sum_{ij\in E} c_{ij}.\left(w_{ij}+s_{ij}\right)+F_{ij}.\delta_{ij} \tag{3.1}$$

Subject to:

1. For each demand, the total source flow equals the demand

$$\sum_{nj\in E} w_{nj}^{r}=d^{r} \qquad \forall r\in D, n=O[r] \tag{3.2}$$

2. For each demand, the total sink flow equals the demand

$$\sum_{jn\in E} w_{jn}^{r}=d^{r} \qquad \forall r\in D, n=T[r] \tag{3.3}$$

3. For each demand, the net flow (incoming – outgoing) for transshipment nodes is zero

$$\sum_{in\in E} w_{in}^{r}-\sum_{nj\in E} w_{nj}^{r}=0 \qquad \forall r\in D, n\notin \{O[r],T[r]\} \tag{3.4}$$

4. Working capacity on an edge must support the simultaneous flows over the edge

$$w_{ij}=\sum_{r\in D} w_{ij}^{r} \qquad \forall ij\in E \tag{3.5}$$

5. For each span failure, the total source restoration flow equals the total affected working capacity

$$\sum_{ik\in E;\, j\neq k} s_{ij}^{ik}=w_{ij} \qquad \forall ij\in E \tag{3.6}$$

6. For each span failure, the total sink restoration flow equals the total affected working capacity

$$\sum_{kj\in E;\, i\neq k} s_{ij}^{kj}=w_{ij} \qquad \forall ij\in E \tag{3.7}$$

7. For each span failure, the net flow for restoration transshipment nodes is zero

$$\sum_{nk\in E:\, k\notin\{i,j\}} s_{ij}^{nk}-\sum_{kn\in E:\, k\notin\{i,j\}} s_{ij}^{kn}=0 \qquad \forall ij\in E, \forall n\notin\{i,j\} \tag{3.8}$$

8. Spare capacity on an edge is at least equal to the largest restoration flow over the edge

$$s_{kl} \geq s_{ij}^{kl}, s_{kl} \geq s_{ji}^{kl} \qquad \forall (ij),(kl) \in E^2, (ij) \neq (kl) \qquad (3.9)$$

9. The edge decision variables can only be 1 (edge included) or 0 (edge absent)

$$w_{ij} + s_{ij} \leq \delta_{ij} K \qquad \delta_{ij} = \delta_{ji}, \delta_{ij} \in \{0,1\} \qquad w_{ij}, s_{ij} = \text{integer} \qquad \forall ij \in E \qquad (3.10)$$

10. Total number of edges is at least equal to number of nodes

$$\sum_{ij \in E} \delta_{ij} \geq N \qquad (3.11)$$

11. Each node is at least of degree 2

$$\sum_{k \in N; i \neq k} \delta_{ik} \geq 2 \qquad \forall i \in N \qquad (3.12)$$

12. The network topology selected is at most of degree $d_{\max}$.

$$\sum_{ij \in E} \delta_{ij} \leq d_{\max} \cdot N/2 \qquad (3.13)$$

Constraints (3.11) to (3.13) may or may not be applied, however they are added knowledge about the desired properties of a qualified network topology to speed up the branch-and-bound solution time of the MIP solver by expressing these desired (and known) topological properties. The model statement for the MTRS problem was taken from [GrDo01].

In this mathematical model of the MTRS problem, we have assumed integral but non-modular capacity units can be placed on spans. In addition, we use a linear model where the capacity cost on a span is a linear function of installed capacity. In reality, transmission capacity systems placed on spans are modular (for example, commercial SONET capacity systems exists as specific multiples of STS-1 units of capacity such as STS-3, STS-12, STS-48, STS-96, STS-192, STS-768) [DoGr00]. Modular capacity placement also offers some economy of scale because the cost-capacity relationship is non-linear. For example, the STS-12 system offers four times the capacity of the STS-3 system at less than four times the cost.

The traditional approach to the problem of capacity placement is to solve the problem for integral values of capacity required on each span, and then round up to the nearest capacity module size available (*post-modular*). On the other hand, we can introduce the modularity into the problem to take advantage of the economy of scale effects (*modular*). [DoGr00] discusses the effect of both design approaches. The required capacity for the post-modular problem was in nearly all cases, higher than that required by the modular problem. In addition, the excess network capacity obtained for the post-modular solutions were found to be higher than that obtained from the modular solutions.

The study also showed that for the modular min-cost design problem, topology reduction is more cost effective and increases with the economy of scale effects despite a mesh network tendency towards higher connectivity to increase the spare capacity sharing efficiency. The explanation offered for this outcome is that "the greater the economy of scale factor and modular capacity available, the more incentive there is for a working or restoration route to detour to help fill a larger module. The modular min-cost solution is characterized by the use of the fewest, largest capacity modules. In contrast, a capacity-minimal solution is dominated by shortest path routing considerations since the min-capacity solution always benefits from greater network average nodal degree solution. Only when we have a linear model (linear cost and non-modular) is the min-capacity cost also the min-cost solution".

When we introduce modularity into the MTRS formulation, the model changes as follows:

Additional parameters:

C_{ij}^{m} is the cost of a module of the mth size on span j.

M is the number of different module capacity systems available.

Z^m is the number of capacity units offered by the mth module size

Variables:

n_{ij}^m is the number of modules of size m placed on span j

The objective is then to minimize the total cost of modules placed in the network:

Modular MTRS:
$$Minimize\left\{\sum_{m=1}^{M}\sum_{ij=1}^{E} C_{ij}^m \cdot n_{ij}^m\right\} + F_{ij} \cdot \delta_{ij} \qquad (3.14)$$

and we add a new set of constraints:

$$s_{ij} + w_{ij} \le \sum_{m=1}^{M} n_{ij}^m \cdot Z^m \qquad \forall ij \in E \qquad (3.15)$$

This additional set of constraints arising from (3.15) ensures that the total working and capacity required on each span would be met by assigning a sufficient number of modules of each size. However, we choose not to introduce the addition complexity of considering modularity in order not to obscure the basic problem of simultaneously optimizing topology, routing and capacity placement.

3.2 THE COMPLEXITY OF THE COMPLETE MTRS PROBLEM

The complexity of the MTRS problem is easily determined from the number of variables and constraints the problem includes. From [GrDo01], the authors show that a graph $G=(V,E)$ with $v=|V|$ and $Y=v(v-1)$ possible unidirectional edges (remember the MTRS problem is cast with $v(v-1)$ unidirectional edge candidates) has $2(Y^2+Y)$ variables and v^4-v constraints. Table 3 shows the number of variables and constraints of the MTRS problem for some node size. We can see the exponential growth of the variables and constraint numbers with problem size. This gives us an appreciation of the need for approximating solutions or simplified decompositions of the full problem.

Table 3 – Variables and Constraints of the MTRS problem for some node sizes

v	Variables	Constraints
6	1860	1290
10	16380	9990
30	1515540	809970
50	12009900	6249950

In [GrDo01], results obtained from the 3-step heuristic were passed to the MIP solver as an upper bound on its solution cost. The authors observed that the solver failed to even reach feasibility (within a given time-limit) when it had an upper bound value whereas, in the absence of the bound, it found feasible solutions. Note that in this case, a feasible solution from a MIP solver is one that satisfied the survivability constraints but not necessarily the connectivity constraints we impose (low connectivity).

We can understand the reason for the solver's inability to reach feasibility from combinatorial principles. There are vastly more arrangements of closed connected graphs with many edges (heavy weight) than there are graphs that have relatively few edges (light weight) that describe a closed connected graph. We illustrate the predominance of heavy weight graph topologies using a 28-node network. For this network, there are 378 possible edges. The total number of edge representations possible for this network size is given by $2^{N(N-1)/2} = 6.2 \times 10^{113}$. This constitutes a very large solution space and can only get bigger for larger problem sizes. The total number of edge representations that can be considered lightweight ($2 \le \overline{d} \le 6$) is $6.2 \times 10^{85} = 10^{-30}$ of the total possible edge representations while the total number of graphs that have $\overline{d} > 6$ is approximately equal to 6.2×10^{113}. The solution space for lightweight edge combinations is infinitesimally small in contrast to solution space for highly connected graphs and therefore explains the MIP solver's inability to escape the dominant regions of heavy weight graphs to regions of the topology space having few edges.

Thus, finding an optimal topology from the complete set of possible edges of an *n*-node graph seems to be the most difficult purely combinatorial aspect of the complete problem. The large combinatorial space of edge combinations that we do not consider to represent qualified graph topologies seems to swamp the ability of a MIP solver to progress towards the goal of finding one closed connected graph on an edge-set that is associated with near optimality [GrDo01].

In Figure 15, we have 3 of the possible solutions for the optimal topology design for a 9-node network. Network (a) shows one of the many possible high-weight solutions a MIP solver would consider. This topology also has stub nodes that do not permit 100% survivability of the network and therefore, should not even be considered. Networks (b) and (c) show potentially good solutions to this problem (Network (b) is the optimal solution obtained from the MIP solver).

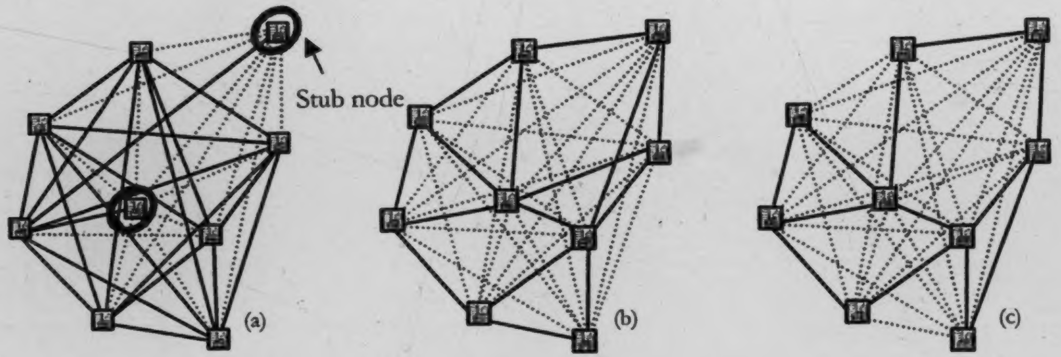


Figure 15 – MIP solutions for 9-node network

Network (a) is not even "plausible" to us, but typifies the sub-optimal solution arising from even 12-hour incomplete MIP MTRS runs suggesting that the development of low-weight feasible graphs is the best sub-problem that we could take away from the MIP model for MTRS. This forms the central hypothesis for our work.

A compelling idea is therefore to limit the graph search space before employing the MIP solver. It seems relatively easy for us to realize what plausible graphs are like, but their discovery is more difficult for the 1/0 MIP solver. The heuristic approaches mentioned earlier all attempt to limit the solution space by proposing candidate topologies for which a detailed or surrogate capacity design is performed. In [GrDo01], the topology solution space is limited for the MIP solver for step J3 to a selection of edge candidates that were considered optimal for routing and restoration purposes.

Another way to limit the solution space, which we will pursue, is to use a topology or graph sifter. The sifter would simply generate feasible low-weight network graphs, which would be passed on to the solver for capacity optimization. With this approach, the solver is guided into a solution space that has been limited to contain fewer edges, a subset of which may be optimal. This should reduce the search or execution time of the optimization process. This approach will be developed in Chapter 5.

Another issue with direct MIP solution attempts for the MTRS problem is that they implicitly have to enumerate a vast number of graphs that have a high edge count since many more high weight combination graphs exist than low weight combinations. This is evidenced by the failure of the solver to find a feasible solution that was similar in cost to (or improved upon) the solution cost obtained from Step 3 when it was provided as an upper bound on the network cost. Routing and spare capacity considerations are carried out before discarding the graphs as being too expensive. This consumes unnecessary resources and would be avoided if certain graph properties could have been detected earlier.

Accordingly, another idea would then be to incorporate a test suite in the graph sifter to detect the presence or absence of desired graph-theoretic properties in advance of even solving the routing and capacity problem. Capacity design would then be carried out only on graphs that possess the properties we deem to be qualified (the graphs that are not rejected by the test suite).

Therefore, a possible approach to solving the MTRS problem would have the following stages:

1. Identify qualified bi-connected graph topologies. The required characteristics of such qualified graphs that we are interested in are:
    a. Is the graph of appropriate weight (not too few nor too many edges)?
    b. Is every node in the graph at least of degree 2?
    c. Is the graph connected?
    d. Is the edge (fixed) cost for the graph lower than some set bound?
    e. Is the graph bi-connected?

    A graph that passes these tests represents a feasible and qualified graph for the transport network.
2. Perform capacity design to determine the routing capacity requirements (and cost) of the network. We could also attempt to solve the restricted MTRS problem for the graph topology if the edge set is sufficiently small.

The benefit of this approach to solving the MTRS problem is that we remove the burden of topology search from the solver and thereby, improve run-time of solution.

4. GRAPH SIFTER METHODOLOGY AND ARCHITECTURE

We have identified that the first step towards solving the reduced MTRS problem would be to generate or obtain low-cost feasible transport graphs by separate means from the 1/0 MIP solver. The graph sifter approach to this involves two steps:

1. **Graph Generation** - Selecting edges from candidate edge set that define the graph,
2. **Graph Testing** - Passing the proposed or generated graph from step 1 through the test suite to determine if it possesses the required characteristics for the optimal network (i.e. is "qualified" for further consideration in terms of routing and capacity optimization).

The third step would be passing the feasible graph from step 2 to the solver for capacity design only, or both capacity and topology optimization (reduced MTRS). In the latter framework, the qualified graph is viewed as giving a small candidate edge set within which to solve a restricted MTRS.

The basic architecture of the proposed graph sifter is as shown in Figure 16 below. The graph generator proposes graphs represented by edge vectors. The tester/sieve checks to determine if the graph is qualified. A graph received at the output of the tester represents a qualified graph topology for which a capacity design can be carried out.

We furthermore want this to be a feedback process so that from any current design, certain information should pass back to the sifter that should affect future solutions. This information could include:

- A best current bound on total network cost, based on the current minimum value found: the edge cost for any proposed topology should not be greater than the lowest total network cost found so far.

- Some metric about the value of the presence or absence of each edge in a solution, how it contributes to the cost of the network. This may enable us to completely eliminate some edges from being considered or to select some edges as being necessary for a good solution.

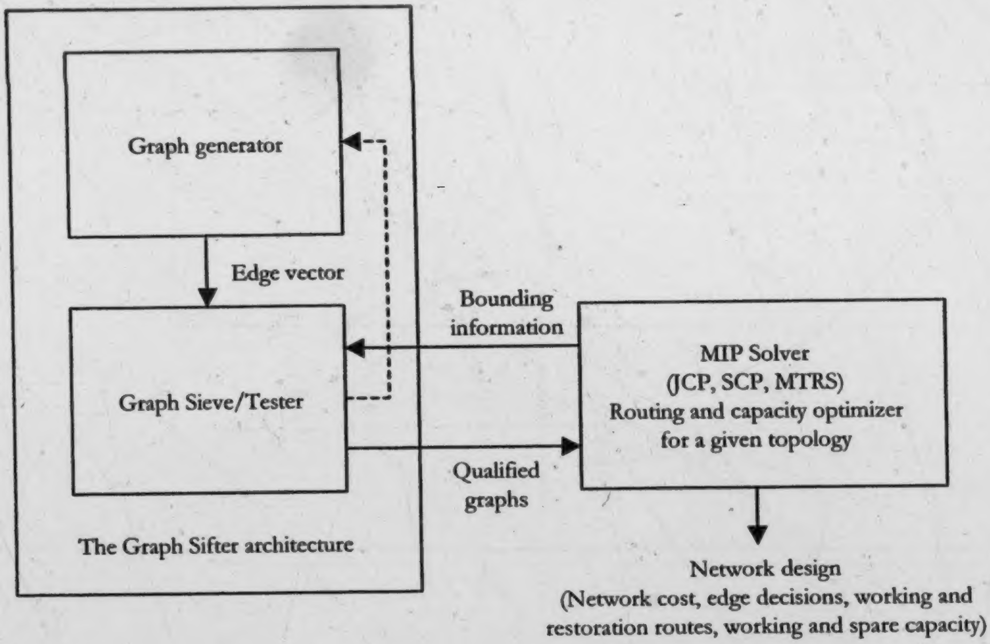

Graph generator
Edge vector
Graph Sieve/Tester
The Graph Sifter architecture
Bounding information
Qualified graphs
MIP Solver
(JCP, SCP, MTRS)
Routing and capacity optimizer
for a given topology
Network design
(Network cost, edge decisions, working and
restoration routes, working and spare capacity)

Figure 16 - Architecture of the proposed graph sifter approach

4.1 GRAPH GENERATION

Given a network with a node-set N, a candidate edge-set E, a graph has an edge vector that is an edge combination based on a binary specification of all the possible edges. From the edges in its edge set E, combined with its edge vector, the edges present in the proposed graph are specified. A 1 in position e of the edge vector means edge e in the edge set is present in the graph, a 0 means it is absent. All tests will involve only those edges specified by the edge vector. For a network with E possible edges, there are 2^E possible graphs. Each base$_{10}$ number between 0 and $2^E - 1$ represents a possible and unique edge combination.

For graph generation or enumeration studies, the question of isomorphism may arise. Two graphs G_1 and G_2 are isomorphic if there is a one-one correspondence between the vertices of G_1 and those of G_2 such that the number of edges joining any two vertices of G_1 is equal to the number of edges joining the corresponding vertices of G_2 [Wils96]. This means that two graph G_1 and G_2 are isomorphic if by re-labeling G_1, we obtain G_2.

The issue of isomorphism can only arise for unlabelled graphs. For example, for the labeled 3-node network, there are eight different labeled graphs but only four unlabelled graphs. Since the transport networks we model are labeled graphs, isomorphism is not a problem. Each edge vector generated represents a unique physical facilities layout when associated with the corresponding edge set.

The next section discusses several graph generation methods we use in this work. Any of these methods can be used when:

- we are feeding generated graphs to the solver,
- we generate an unbiased randomly chosen starting topology for any of the heuristics.

We now present the methods we use to generate possible graphs of the network topology:

4.1.1 SEQUENTIAL GRAPH GENERATION METHOD

In the sequential generation approach, the edge vector of the graph is a binary representation of a base$_{10}$ number. We start from 0 and generate graphs with the edge combination specified by each base$_{10}$ number up to $2^E - 1$. If run to completion, this would enumerate a total of 2^E distinct graphs having all possible edge combinations. Each graph generated is tested. If a graph G passes the first test, it undergoes the second test and so forth.

However, the purely sequential generation of all possible graphs for the network poses a problem of time complexity. Table 4 illustrates the increase in the time complexity with increase in number of nodes in the network size and shows that at a (still optimistic) test rate of one thousand (1000) graphs per second, it would take over two years to evaluate all possible graphs of a 9-node network and over 11 centuries for the 10-node network!

Table 4 – Time complexity with network size

Node Size	Number of edges (universal set)	Number of possible graphs	Time at evaluation rate of 1000/sec
7	21	2.10×10^{6}	~35 minutes
8	28	2.68×10^{8}	~75 hours
9	36	6.87×10^{10}	~2.18 years
10	45	3.52×10^{13}	> 1115.7 years

Clearly, it would not prove feasible to evaluate all possible edge combinations for a graph on N nodes where N is of interesting practical sizes.

In Chapter 5, we conduct some test studies and our results show that problems with $|E| > 21$, it becomes very difficult and can be considered impractical, to generate and test all possible edge combinations with this method. Thus, it may be used only for graphs having small node set ($|N| \leq 7$) when all possible spans are being considered. For graphs with $|N| \geq 8$ or $|E| > 21$, a random or a deterministic generation approach will have to be used. The graph sifter is still quite a useful development however, for qualifying randomly generated samples on these larger problems.

The next section deals with the random graph generation models we use as an alternative to the sequential graph generation approach.

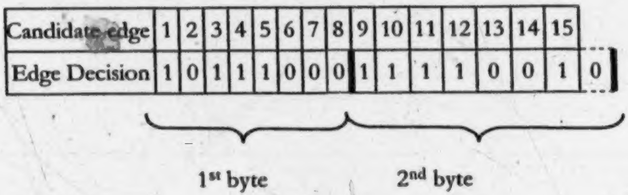
4.1.2 RANDOM GENERATION METHODS

Following from the sequential generation methods, the first obvious random generation method is to generate random numbers in the range $[0, 2^E]$ and the edge vector would then be the binary equivalent. The limitation to this is that in Java (the implementation language used in this work), the largest integer data type (type *long*) only models the set of integers from approximately –8 quintillion to 8 quintillion (1 quintillion = 10^{15}). For our purposes, this represents a graph having less than 53 edges (the full edge combinations for an 11-node, 55-span network can not be realized). Therefore, we have to use some other random graph generation methods.

The random graph generation methods presented in the following sections construct a graph by adding edges to a given set of nodes with a probability p. The differences in the methods arise from how p is determined and whether p influences the addition of a single or multiple edges. The random graph generation techniques we use are:

4.1.2.1. Byte-wise generation

For the byte-wise generation, the edge vector is partitioned into groups of eight (8) slots each, which are equivalent to bytes. For each group, an integer value in the range [0,255] is randomly generated then encoded as 8 bits. The value of each bit in the byte is the value of each slot position in the group. For the last set of slots, which may not be up to 8 slots, say m slots, the higher order m bits of the byte value are used. A concatenation of all the bytes defines the edge vector as shown in the figure below.



Candidate edge	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Edge Decision	1	0	1	1	1	0	0	0	1	1	1	1	0	0	1	0

1 = edge present
0 = edge not present

4.1.2.2. Bit-wise generation

For all bit-wise generation methods, each edge is considered and is added to the graph with a probability p. For each slot in the edge vector, a random number in the range [0, 1] is generated. If the number is less than or equal to p, the slot is set to 1, else, it is set to 0. The function used to determine p leads to the different models for generating random graphs. For this work, we have used the following bit-wise graph generation models:

a. Pure Random Probability Model

p is a fixed edge probability. Here, the expected or desired maximum nodal degree is defined at the onset and the edge probability is determined from that. For example, suppose we want a graph having average nodal degree of 5 for a 10-node network. This represents a graph with 10 nodes and 25 edges, out of possible 45 edges. The edge probability p would then be set at $25/45 = 0.56$.

The fixed probability model does not reflect the structure of real network topologies because it does not use any locality information about the graph in making edge decisions. Other random generation methods presented use a probability function that depends on the distance (Euclidean) between the nodes. This is an attempt to reflect the structure of real networks by reducing the probability of having long edges in the graph.

b. Waxman Random Graph Model

In this model, the probability of an edge between two vertices *u* and *v*, is given by:

$$p(u,v) = \alpha e^{-d/\beta L}$$

where $0 < \alpha, \beta \leq 1$ are parameters of the model, d is the (Euclidean) distance from u to v, and L is the maximum distance between any two nodes in the plane of the network problem. An increase in α and β increases the probability of the edge (u, v) and increases the number of edges in the graph. More important is the relative function of each parameter: An increase in α increases the propensity for shorter edges while an increase in β will increase the ratio of long edges relative to shorter edges [ZeCB96].

For a transport network graph, we want a higher ratio of shorter edges to longer edges. Thus, we set $\alpha > \beta$. We do not have an analytical expression from which we can derive values of α and β. However in [ZeCB96], empirical studies were carried out to determine the values of α, β which give graphs with a fixed nodal degree of 3.5. The values chosen were $\alpha = 0.2$, $\beta = 0.15$ out of many combinations of the parameters that achieve this target. We select the values for these parameters that we use in this work from empirical studies with various combinations of α and β to find approximate values of these parameters that would increase the probability of obtaining a connected random graph. We chose $\alpha = 3.5$, $\beta = 0.5$.

c. Exponential Random Graph Model

This model was proposed in [ZeCB96] and related the edge probability to the distance between vertices using more direct probability functions compared to the Waxman model (uses one parameter (α) whereas the Waxman model uses two (α, β)). The probability of an edge between two vertices *u* and *v*, is given by:

$$p(u,v) = \alpha e^{-d/L-d}$$

d is the (Euclidean) distance from u to v, and L is the maximum distance between any two nodes in the plane of the network problem. The probability of an edge approaches zero as d approaches L and the expected average nodal degree scales linearly with α. We set the value of α to 3 based on results of empirical studies.

In [ZeCB96], the authors showed that the probability of producing a graph that is connected or closed from entirely random selections is minimal. This is verified by our tests (in Chapter 5) on graphs constructed using the random generated techniques. The tests showed that these methods have very low probabilities of producing connected graphs. This leads us to search for deterministic (algorithmic or heuristic) methods that we would use to select the required *k* edges from the set of *E* edges and have a higher probability of being connected. These methods are described in the following section.

4.1.3 ALGORITHMIC GRAPH GENERATION METHODS

Algorithmic graph generation methods provide a way for us to generate graphs that are at least connected. However, such graphs may not satisfy the survivability constraints. The methods we use are described below:

4.1.3.1. Node Linking

A vertex *n* is selected at random from all nodes in the graph as the start node and is added to the record of nodes visited. At *n*, the cheapest (shortest) candidate adjacent edge is added to the graph. The adjacent vertex *v* for the added edge becomes the next node to be visited and added to the record. At *v*, the cheapest candidate adjacent edge not yet in the graph is added. This process is continued until the graph is connected (when the record of nodes visited contains all nodes in the graph).

A tie occurs if the edge selected from a vertex *v* has its other vertex *n* already 'linked' in the graph or vertex *n* has all its candidate adjacent neighbors already linked. To break a tie, we choose another vertex *u* randomly from the linked vertices such that *u* has a candidate adjacent vertex *w* not yet linked. From vertex *w*, the nodal linking process continues. We do this in an attempt to ensure the graph is connected with a minimum number of edges.

4.1.3.2. Minimum Spanning Tree Model

A minimum spanning tree for the graph is found using the Prim's algorithm outlined earlier. Like the Node Linking method, we obtain a definitely connected graph from this process with N edges. The resulting tree can be made survivable by applying a repair heuristic to make it two-edge or two-node connected.

4.1.3.3. Two Trees

This graph generation method is the only graph generation method that produces a qualified candidate without the need for applying a repair heuristic. It is a variant of the Two-Trees Dense graph construction heuristic described in [MoSh89]. The method involves combining two trees formed on graph $G=(V,E)$ to form a graph $G''=(V,E'')$. The first tree is a minimum cost spanning tree $T=(V,E')$ spanning V. The second tree T' is formed on graph $G'=(V,E-E')$. The variation between the method we employ in this work and that described in [MoSh89] is how T' is chosen.

In [MoSh89], the problem studied involved two types of nodes having different survivability requirements (special nodes need a two-node connected network, while other nodes need just a connected network). In this work, all nodes have the same survivability requirements (two-node connectivity). We choose the second tree as a minimum cost spanning tree formed on the graph $G' = (V, E - E')$ with no edge of T re-used. This method generates two-node connected (but dense) networks and is illustrated in Figure 17. In Figure 17(a), the solid lines show a minimum spanning tree T for the 7-node, 21-span network. Figure 17(b) shows a minimum spanning tree T' for the graph after the edges of T are removed. Figure 17(c) shows the resultant graph $G'' = (V, T \cup T')$.

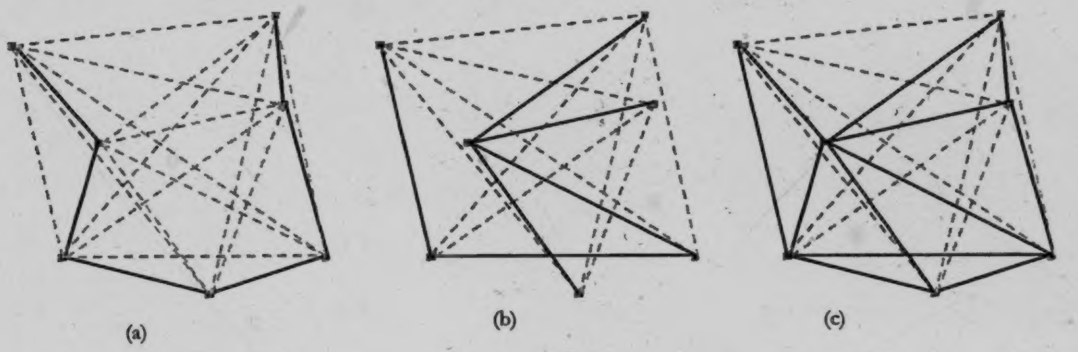


Figure 17 - Two Trees graph generation

The various graph generation routines are summarized in Figure 18.

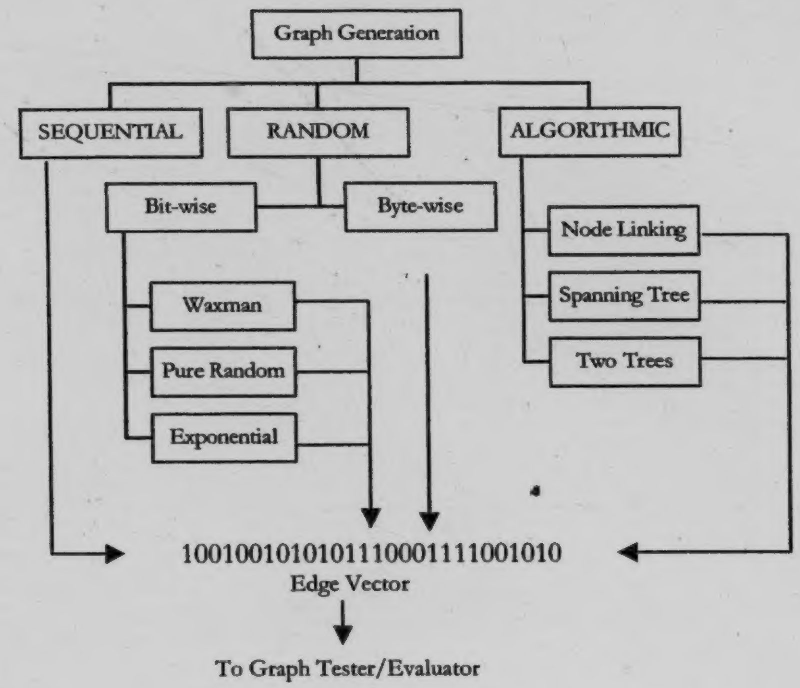

Graph Generation
SEQUENTIAL
RANDOM
ALGORITHMIC
Bit-wise
Byte-wise
Waxman
Pure Random
Exponential
Node Linking
Spanning Tree
Two Trees
1001001010101110001111001010
Edge Vector
To Graph Tester/Evaluator

Figure 18 – Methods for graph generation

4.2 GRAPH TESTING

The set of test criteria is made up of the following six tests:

4.2.1 TEST 1 - TEST FOR MINIMUM AND MAXIMUM EDGE COUNT

A bi-connected graph will *always contain at least* N *edges*. Thus a minimum number of edges for a qualified graph is N. There is also a maximum edge weight disqualifier since we do not need high edge weight graphs to satisfy survivability constraints. Following from our look at real networks, we saw that the maximum average nodal degree for the topologies to be $d_{\max} < 4.5$. For the purposes of our study, we use the more generous value of $d_{\max} = 5$ and the maximum number of edges we allow for a graph is $k \le d_{\max} \cdot N/2$.

This first and simplest test checks that the relationship $N \leq k \leq d_{\max} \cdot N/2$ is satisfied. If the edge count k, of graph G satisfies the above relationship, G has passed Test 1 and goes on to Test 2. Otherwise, it is discarded and a new graph candidate generated.

4.2.2 TEST 2 - TEST FOR STUB NODES

A bi-connected graph *has no node with degree less than 2*. Each node in the graph G will be individually checked for the condition that $\deg(n) \geq 2$ is satisfied, where $\deg(n)$ is the degree of node n. If all nodes in G meet this condition, G passed Test 2 and goes on to Test 3. Else, G is discarded and a new graph candidate generated.

4.2.3 TEST 3 - TEST FOR A CONNECTED GRAPH

A connected graph is one that cannot be expressed as a union of two graphs. It has a single component, such that there is a path between every two vertices. Thus from any node, all other nodes in the graph must be accessible. If this is true, then all nodes are accessible to each other, at least through that node.

We test for this property with the graph traversal routine earlier presented. Starting at any node in G, which we will call the *root node*, we call the graph traversal routine described earlier. When all connected nodes have been visited, the visited node record is checked to ensure that it contains all nodes in the graph. If it does, G had passed Test 3 and goes on to Test 4. Otherwise, it is discarded and a new graph candidate generated.

4.2.4 TEST 4 - TEST FOR MINIMUM COST GRAPH

Test 4 checks that the sum of the fixed cost of edges included in the graph are below some maximum edge cost budget. The purpose of this test is to eliminate graphs with high edge costs before we expend computing resources in more involving test (Tests 5 and 6). This is a similar test to Test 1, but yet different. Test 1 ensures that the connectivity of the graph is within the desired connectivity range while Test 4 checks that the edge cost of the graph is below a certain cost. Two graphs could have the same connectivity with different edge costs depending on the weight (or costs of the edges included). The bounding fixed cost can be an input from the MIP solver on the best-cost network seen so far.

If G has its fixed cost within the budget, it is passed to Test 5. Otherwise, it is discarded and a new graph candidate generated.

4.2.5 TEST 5 - TEST FOR TWO-EDGE CONNECTED GRAPH

A qualified transport network should have at least two span-disjoint paths between any node pair. Thus, the implication of this requirement for our purposes is that the graph must be at least, *2-edge connected*. To this effect, we require graphs having $\lambda(G) \geq 2$. A graph that is not 2-edge connected contains a cut-set with only one edge – the bridge. Test 5 checks the graph G for the existence of a bridge using the algorithm described earlier for testing to-connectivity. If none exists, G has passed Test 5 and goes on to Test 6. Otherwise, it is discarded and a new graph candidate generated.

4.2.6 TEST 6 - TEST FOR BI-CONNECTED GRAPH

A graph that is not two-node connected contains a separating set with only one node, the cut vertex. The number of bi-components in G signifies the presence or absence of a cut vertex. Test 6 checks the graph G for the existence of a cut vertex using the two-connected algorithm described earlier. If a cut-vertex is found (multiple bi-components), G is discarded and a new graph candidate generated. Otherwise, G represents a qualified graph for further optimization.

These tests are administered in the order presented which is also the order of increasing importance and computational expense required. Only graphs that are not rejected by the prior test are tested by the next test. No repair is actually carried out at any stage in the graph sifter. Candidate edge specifications that are not rejected by the entire graph sifter test suite are then passed to a CPLEX solver for fast mesh routing and capacity optimization. Further optimization may also involve passing the graph through a restricted MTRS solver for final topology selection and joint capacity allocation. The result would be a sub-graph of the graph proposed with optimized working and restoration flow routing and capacities to realize a minimum cost network.

Figure 19 shows the flow diagram for the test suite for one instance of graph generation and testing.

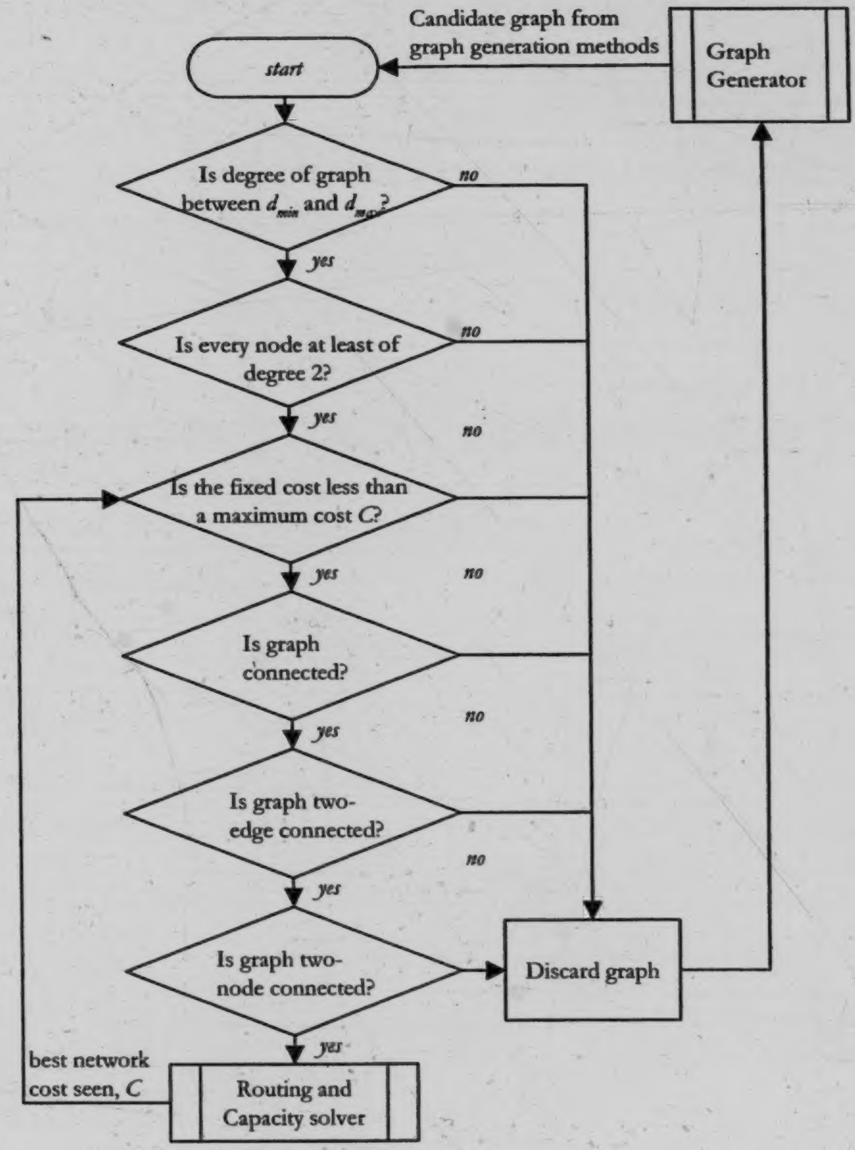

Candidate graph from
graph generation methods
Graph
Generator
start
Is degree of graph between d_{min} and d_{max}?
no
yes
Is every node at least of degree 2?
no
yes
no
Is the fixed cost less than a maximum cost C?
yes
no
Is graph connected?
yes
no
Is graph two-edge connected?
no
yes
Is graph two-node connected?
Discard graph
yes
best network cost seen, C
Routing and Capacity solver

Figure 19 – The Graph Sifter

We have presented a large variety of graph generation methods that could be used in our studies. The specific choice of graph generator depends on the intended purpose of the graph. For the statistical studies presented in Chapter 5, we used the sequential generation approach because our goal is to enumerate all possible edge combinations. In Chapter 6, we present two heuristics – Iterative Sampling and Sweep Search. For the Iterative Sampling heuristic, we found the rate of convergence on a near-optimal solution was improved if the minimum spanning tree generator was used. The repair heuristic then makes it two-node (or –edge) connected. For the Sweep Search heuristic, the starting topology is required to be dense. Therefore, any of the graph generators can be used and the repair heuristic is applied to the generated graph to ensure the required degree of connectivity and survivability.

We employ a repair heuristic involving a sequence of steps (edge additions or removals) to imbue any graph with the graph theoretic properties of interest. The repair heuristic is made up of specific routines that deal with a particular graph property. For example, if a graph fails Test 1 (test for maximum or minimum span count), the specific routine to repair the graph would be to add or remove edges to/from the graph respectively, until the weight of the graph is within the desirable range. If the graph fails Test 2, we would add an adjacent span to each node with degree ≤ 1. For Tests 3, 5 and 6, the repair routine first finds the components in the graph. Then an edge is added between nodes in different components. The respective test is repeated and new components (if any are identified). Edges are added between nodes in different components and this process is repeated until the graph is connected, two-edge or two-node connected. For Test 4, the repair routine exchanges high cost edges with lower cost edges.

The repair process can be applied to any graph generated using any of the graph generation methods. However, graphs generated from the random generation methods are usually denser than graphs obtained from the algorithmic methods after repair. This is typically because the edges selected by the random method do not produce a connected graph hence, more edges are required to be added to the graph to make it bi-connected (repair), whereas the algorithmic process generates sparser networks and fewer edges may be required for the repair process. Note that the repair heuristic is only applied to a graph if one feasible topology is required.

The test suite may be coupled to the repair routine process such that no graph is actually discarded but adapted to suit a qualified topology. This is only necessary when we want a starting feasible topology solution for applying any of the heuristics. The coupled module proved to be advantageous for a number of reasons:

1. The only graph generation method that is guaranteed to construct a two-connected graph is the two-trees method. Other methods would require numerous graph constructions before we may be able to obtain a graph that is qualified.
2. The space and time complexity of repairing a graph is probably less than that required to generate a number of graphs before obtaining one that is both lightweight and two-connected.
3. Once we have an initial or starting qualified graph, we are easily able to obtain similar qualified graphs by addition, removal or exchange of edges.

Figure 20 shows the flow diagram for the graph sifter with stage-by-stage repair coupled module.

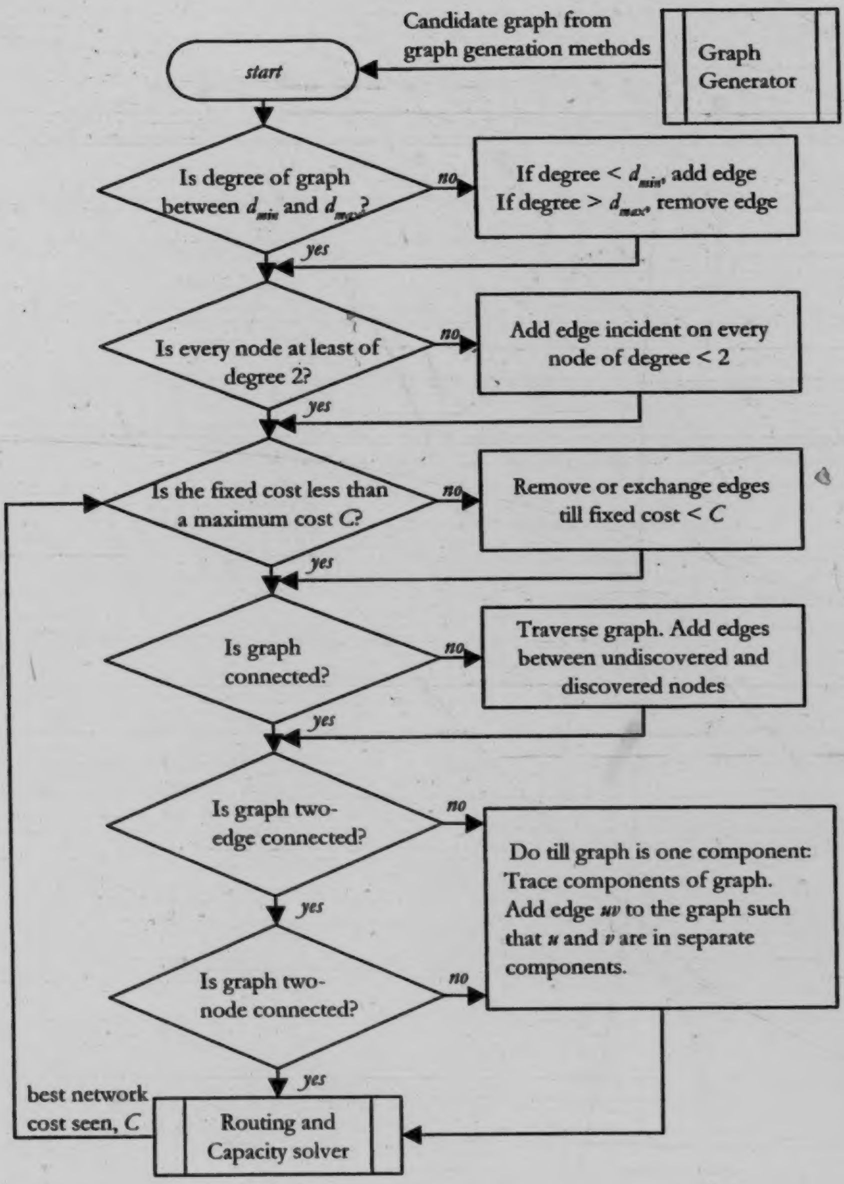

Candidate graph from graph generation methods
Graph Generator
start
Is degree of graph between d_{min} and d_{max}?
no
If degree < d_{min}, add edge
If degree > d_{max}, remove edge
yes
Is every node at least of degree 2?
no
Add edge incident on every node of degree < 2
yes
Is the fixed cost less than a maximum cost C?
no
Remove or exchange edges till fixed cost < C
yes
Is graph connected?
no
Traverse graph. Add edges between undiscovered and discovered nodes
yes
Is graph two-edge connected?
no
Do till graph is one component: Trace components of graph. Add edge uv to the graph such that u and v are in separate components.
yes
Is graph two-node connected?
no
yes
best network cost seen, C
Routing and Capacity solver

Figure 20 – The Graph Sifter coupled with the optional repair process

Figure 21 shows the overall flow diagram for the graph sifting network design approach using the graph generation and testing modules.

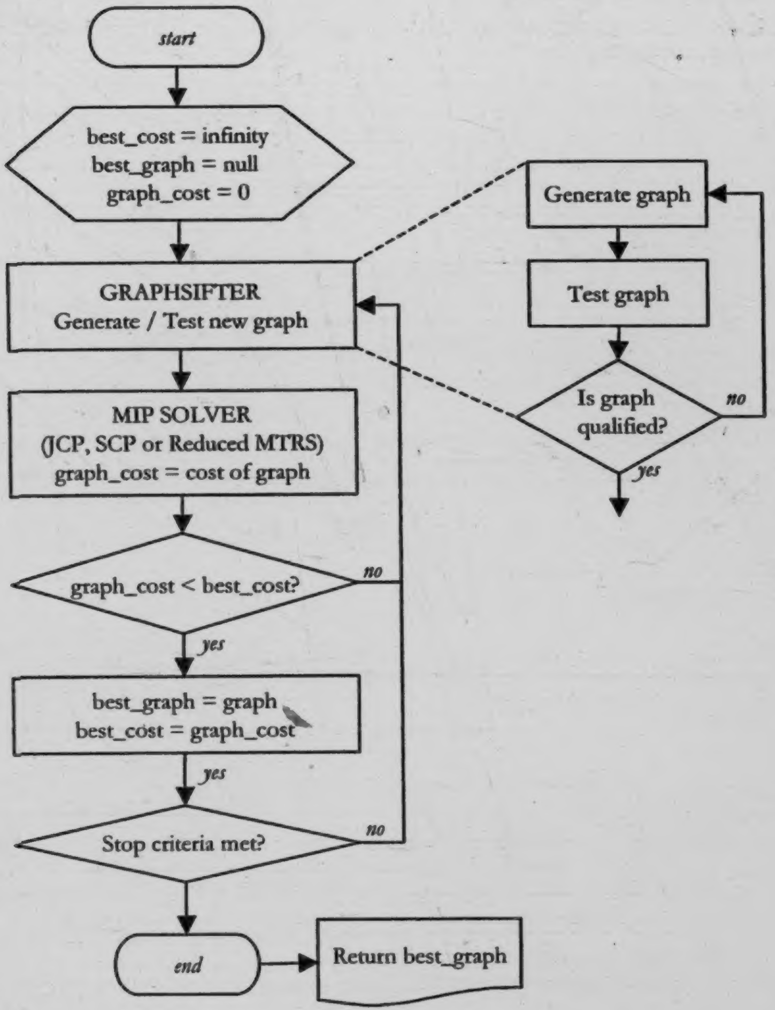

start
best_cost = infinity
best_graph = null
graph_cost = 0
GRAPHSIFTER
Generate / Test new graph
Generate graph
Test graph
Is graph qualified?
no
yes
MIP SOLVER
(JCP, SCP or Reduced MTRS)
graph_cost = cost of graph
graph_cost < best_cost?
no
yes
best_graph = graph
best_cost = graph_cost
yes
Stop criteria met?
no
end
Return best_graph

Figure 21 – Flow diagram illustrating the composite graph generation and sifting modules

5. PRELIMINARY STUDIES

5.1 COMPUTING ENVIRONMENT

In the thesis, we used two CPLEX MIP solver versions running on different computing platforms for solving the optimization problems. One is the CPLEX 7.5 MIP solver on a uni-processor platform (AMD Athlon™) running the Windows 2000 operating system at 1.2 GHz with 256 MB RAM. The second is CPLEX 7.1 MIP solver on a multi-processor platform (4x450MHz) Sun Enterprise 420R server running the Solaris 8 operating environment with 4 GB RAM. When we use the uni-processor platform, the reported run time is the real elapsed system time. When we use the multi-processor platform, the reported run time is the CPU seconds (not the real elapsed time). The CPU seconds gives the total amount of time each of the processors devoted to the task and therefore, is comparable to the amount of time a uni-processor system would take to perform the same task. The graph sifter program is implemented using Java and is run on the uni-processor computing platform.

The COST239 network is a published network of major European cities with 11 nodes and 26 spans. From this network, we made a test case with complete edge set (55 spans). The lengths of the additional spans were obtained by scaling the Euclidean length of the span by an integer factor. This factor was obtained as the average over such scaling factors (ratio of the given span length to its Euclidean length) for the 26 spans in the original network. From this complete 11-node network, we derived a family of COST239 sub-networks (10-nodes, 9-nodes, 8-nodes, 7-nodes and 6-nodes) by removing the appropriate number of nodes from the network. The networks are listed in Appendix B and are named in the form – COST239-*X*n network where *X* is the number of nodes.

The COST239 network also has a published demand matrix that is complete (all node pairs exchange non-zero demand units) but not all uniform quantities. We obtained the demand matrix for the COST239 family of networks again by removing the demands for nodes that are not in the corresponding sub-network.

Additional test problems used were obtained from the study conducted in [GrDo01]. The network data and demand patterns used can be found in the Appendices A and B respectively.

5.2 STATISTICAL STUDIES USING THE GRAPH SIFTING APPROACH

Now that the basic functions and capabilities of the graph generator and sifter modules have been developed, we evaluate the performance of using these modules in feeding the optimization solver. We wish to be able to answer the following questions: just how big is the problem we wish to solve? How small a needle are we seeking in the haystack? Thus, this initial test phase serves as a learning exercise.

First, we wish to investigate the probability of having a 2-connected graph on any given node set and we do this in two stages. For both stages, we employ the sequential graph generation method. For each test, a count is kept for the number of graphs that were tested and the number of qualified candidates. Test 4 (test for maximum fixed costs) is excluded because its relevance comes when the routing and capacity design considerations are considered. These tests are solely for the purpose of obtaining a statistical summary of the probabilities of a generated graph having any of the graph-theoretic properties of a qualified transport network graph.

In the first stage of the graph generation and testing, the candidate edge set considered is the universal set $E = N(N-1)/2$ of all the possible spans for an N-node network. The number of nodes N for the network under consideration is the only given parameter. The statistical results obtained are shown in Table 5 below. The table shows the number of graphs generated and the number that passed each test.

Table 5 - Test Suite Results (universe of all possible edges used)

$\|N\|$	2^E graphs	Test 1	Test 2	Test 3	Test 5	Test 6
4	64	22	10	10	10	10
5	1024	638	253	253	253	241
6	32,768	27,824	12,068	12,058	11,968	11,468
7	2,097,152	2,014,992	1,052,793	1,052,443	1,047,613	1,019,473

For the 8-node network with 28 edges, 2^{28} = 268,435,456 distinct graph combinations exist. However, 48 hours of running the program did not exhaust the solution space and the program was then terminated.

In the second stage, the edge set considered is a subset of the universal set, having only selected edges from E. The smaller candidate edge set reduced the solution space, and made it possible for 8-, 9-and 10- node networks to be considered. Results obtained are as given in Table 6 below. The name of the test case (e.g. 9n17s1) indicates the size of nodes (9), the size of its candidate edge set (17) and a number (1) that distinguishes between problems of same size. The network data is given in Appendix A.

Table 6 – Test Suite Results (subset of possible edges used)

Test case	2^E graphs	Test 1	Test 2	Test 3	Test 5	Test 6	Duration
9n17s1	131,072	65,536	4,596	4,554	4,144	3,289	0:1:08
9n17s2	131,072	65,536	1,920	1,906	1,759	1,500	0:0:37
10n19s1	524,288	262,144	11,677	11,579	10,489	7,908	0:4:59
10n19s2	524,288	262,144	19,222	18,964	16,888	15,167	0:7:52
10n19s3	524,288	262,144	10,975	10,849	9,941	7,836	0:4:59

Tests involving over 21 spans have proven to be difficult to solve to a timely completion using the sequential generation method. We also realize that these results are not typical for the problem size, say 10 nodes and 19 spans. A different selection of edges to be considered may yield different statistics.

OBSERVATIONS

We can make some interesting observations from the test results:

1. If the number of edges E to be considered for an N-node network is 2N-1 for which the number of possible graphs 2^{2N-1}, the number of graphs having span count greater than or equal to N is exactly 2^{2N-2}.
2. For a network with N nodes and a candidate edge set E, the number of graphs with exactly N spans is ${}^{E}C_{N}$. Therefore, the number of graphs G_N of the possible 2^E graphs with a span count greater than or equal to N, is given by

$$G_N = {}^{E}C_{N} + {}^{E}C_{N+1} + {}^{E}C_{N+2} + {}^{E}C_{N+3} + ... + {}^{E}C_{E}$$

3. Similarly, the number of graphs with less than N spans is $2^E - G_N$ expressed as,

$${}^{E}C_{0} + {}^{E}C_{1} + {}^{E}C_{2} + {}^{E}C_{3} + ... + {}^{E}C_{E-1}$$

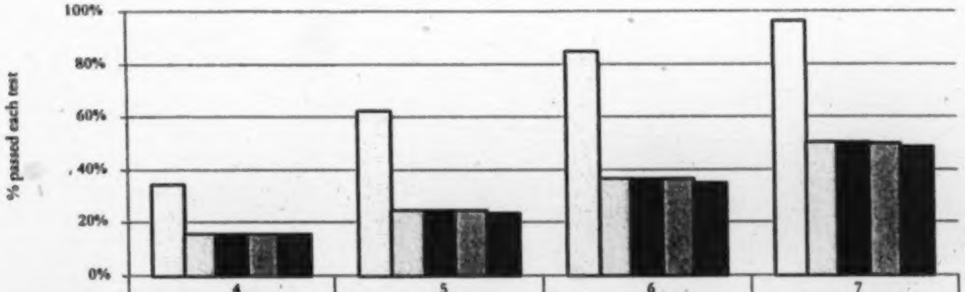
4. When we then add the maximum edge weight disqualifier, we can obtain the number of graphs which would have an edge count between the minimum and maximum edge weights given as:

$${}^{E}C_{N} + {}^{E}C_{N+1} + {}^{E}C_{N+2} + {}^{E}C_{N+3} + ... + {}^{E}C_{X}$$

   where X and N are the maximum and minimum edge weights respectively.

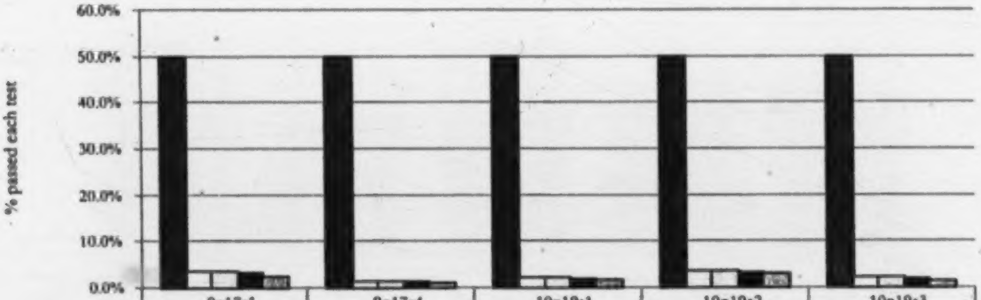
5. We observe that the greatest sifting of unqualified graphs happens at Test 2. Between Tests 2 and 6, approximately more than 80% of the candidate graphs that passed Test

2 will pass Test 6 and are therefore, bi-connected. This is depicted in Figure 22 and Figure 23.



	4	5	6	7
☐ Test 1	34.4%	62.3%	84.9%	96.1%
☐ Test 2	15.6%	24.7%	36.8%	50.2%
■ Test 3	15.6%	24.7%	36.8%	50.2%
▨ Test 5	15.6%	24.7%	36.5%	50.0%
■ Test 6	15.6%	23.5%	35.0%	48.6%

Figure 22 – Summary of graph qualifications with generation given complete edge set



	9n17s1	9n17s4	10n19s1	10n19s2	10n19s3
■ Test 1	50.0%	50.0%	50.0%	50.0%	50.0%
☐ Test 2	3.5%	1.5%	2.2%	3.7%	2.1%
☐ Test 3	3.5%	1.5%	2.2%	3.6%	2.1%
■ Test 5	3.2%	1.3%	2.0%	3.2%	1.9%
▨ Test 6	2.5%	1.1%	1.5%	2.9%	1.5%

Figure 23 – Graph qualifications with generation given subset of universe of edges

The trend in the outcome of these tests leads us to the third test phase. We want to verify that even for large problem sizes, Test 2 is a major sifter. To do this, we used the byte-wise random graph generator and generated one million instances of networks with node size as shown in Table 6. The eligible edge set for the graphs is the universal set, containing all the possible edges.

Table 7 –Test results for a sample of random graphs on larger N

$\|N\|$	Sample size	Test 1	Test 2	Test 3	Test 4A	Test 4B	Duration
8	1,000,000	993,791	630,844	630,786	629,666	620,879	0:03:14
12	1,000,000	805,368	740,089	740,089	740,082	739,809	0:10:17
16	1,000,000	17,563	16,340	16,340	16,340	16,339	0:22:10
20	1,000,000	0	0	0	0	0	0:50:43
24	1,000,000	0	0	0	0	0	1:41:22
28	1,000,000	0	0	0	0	0	3:09:42

It is interesting to note that as the size of the node set increases, the probability of directly generating a qualified graph (i.e. one that does not need repair) dwindles to nearly zero. More specifically out of one million random samples, no qualified graphs were produced at all! For N = 20, 24, 28, not even one graph with simply the right edge weight was produced. As N increases, the probability of getting a graph with the desired edge count decreases to zero. In Chapter 3.2, we illustrated the complexity of solving the full MTRS problem using a 28-node network. For this network, we saw that the total number of edge representations that are lightweight ($2 \leq \bar{d} \leq 6$) and can be qualified transport networks is $6.2 \times 10^{85} = 10^{-30}$ of all possible edge representations. What then is the probability that from one million random samples, a bi-connected graph would be generated in these test cases? For all practical purposes, it is *nil*.

Therefore, we see that for large problem sizes, neither sequential nor random generation techniques would suffice. We need some algorithmic or heuristic means of selecting the *k* edges that make up a graph from the edge set under consideration. For $|N|$ =8, 12, 16, Test

2 still proves to be the major sifter. It would seem quite reasonable to assume that for larger problem instances, the same would hold true.

Based on practical considerations, we decided that the most important property we want in a qualified network topology was two-node connectivity (i.e. bi-connectivity). Therefore, in further work, we carry out Test 6 immediately after Test 4 (instead of both Tests 5 and 6) to obtain a decrease in the program execution time.

5.3 EMPIRICAL STUDIES OF THE EFFECT OF Ω ON TOTAL NETWORK COSTS

This is another section reporting preliminary studies that were conducted to gain understanding of the MTRS problem as a whole. In this section, we study the effects of the various parameters that influence the total cost of network. Specifically, we study the effect of demand and fixed cost variation on the total network cost.

In Equation (2.2), we saw more directly the role of Ω and $(w_i + s_i)$ on C. We can then say that $C = f(\Omega, D)$, two varying parameters. To be able to study the direct effect of both D and Ω on C, we keep one parameter constant and study the other, then repeat with the second parameter. In what follows, we look at how each of these impacts on the network topology design and total network costs starting with the effect of Ω on C with constant D.

When the fixed cost is negligible or small, the total network cost is dominated by the capacity cost. We know that the capacity costs reduce with increase in the connectivity of the network. As a result, the optimal network (for this case of relatively low fixed cost) pushes towards a well-connected mesh network with high average nodal degree. In the same vein, when the network fixed costs dominates, the optimal network pushes towards a single Hamiltonian ring network.

In reality, we do not have any control over the edge establishment costs. They are determined by the goods and services required for the acquisition of land/right-of-way, excavation, trench digging, etc. Thus, the relationship between the edge and capacity costs would probably vary with each edge in the network and from network to network. One edge may be a leased right-of-way from another carrier at normal cost whereas another might require trenching through the Canadian Shield (Pre-Cambrian rock). For the purposes of our research however, we need to establish an empirical value to depict the edge-to-capacity cost ratio (Ω) ratio. The edge costs can only appreciate as the value of land appreciates and the cost of labor increases while transmission costs per bit continue to reduce through technology advances. Thus the value of Ω would seem to see an increase whereas capacity costs go down with technology advances in DWDM.

We do not know the real world value of the cost ratio Ω. However for research purposes, we choose to study the MTRS problem within a region where capacity and edge establishment costs are of significant importance. This is the reason why we need to set the value of Ω carefully. Let us now treat Ω as a study parameter to cover a range of cost ratios of establishment to capacity costs.

The fixed cost for establishing an edge would be $l_{ij}\Omega$ and the unit capacity cost for the edge is l_{ij}, where l_{ij} is the length of span (i, j). Ω can be defined to mean the number of capacity units established on an edge that are equivalent to the cost of the edge. A Ω value of 20 would then mean that for each edge, 20 units of capacity placed on the span (i, j) has the same cost as establishing the edge in the first place. Less than 20 units of capacity on the edge means that the fixed cost dominated, while more than 20 units signifies that capacity costs dominate the total cost incurred for that edge.

To study the effect of varying Ω values, we determined the cost-optimal network by solving the formal MTRS problem model (the AMPL model is included in Appendix C) to optimality with the universal edge set for the COST239-6n, COST239-7n and COST239-8n networks using the COST239-6n, COST239-7n and COST239-8n demand patterns respectively. Similar results were obtained in all three test cases so we present only the COST239-8n test results.

We vary the connectivity of the graph (by adding and removing edges) and solve the routing and capacity optimization problem on the resulting graph. The plot of the total network costs versus connectivity is similar to Figure 8 (increase in connectivity brings about an increase in edge establishment costs while reducing capacity costs). This reinforces the basic hypothesis that an optimal topology exists for which the cost of provisioning capacity and the sum total of all edge costs in the network is minimum.

Figure 24 shows the plot of total network cost obtained from the solution of the formal MTRS problem for network COST239-8n with varying Ω values. From this plot, we see that an increase in Ω contributes to increasing the absolute network cost though we may be driving the topology through the region of relative (edge/capacity) cost that would correspond to a true cost minimum topology (the region within which we believe the optimal topology to lie).

Within some range of values of Ω, the optimal graph topology does not change with increase in Ω (easily seen in Figure 24 by the data points at a plateau. For example, within the region $\Omega = 45 - 60$) and therefore in these regions, the amount of capacity (and the cost) required to serve the demands is same because the capacity cost is influenced solely by the physical topology.

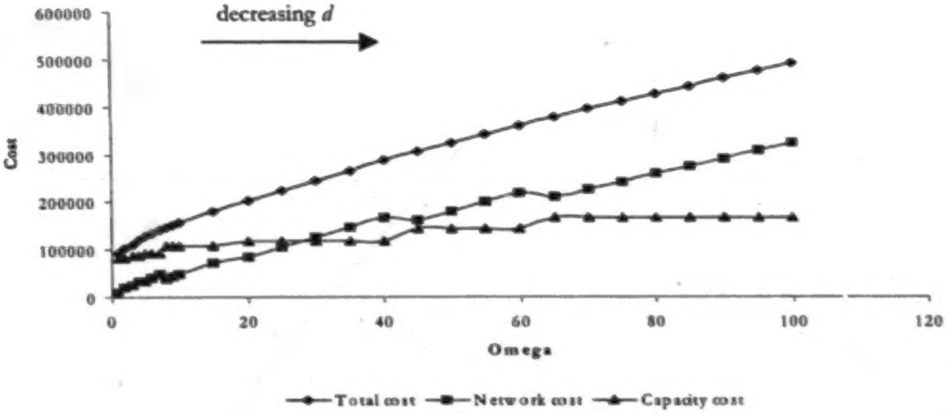


Figure 24 – Total cost of optimal MTRS solution vs. Ω for network COST239-8n

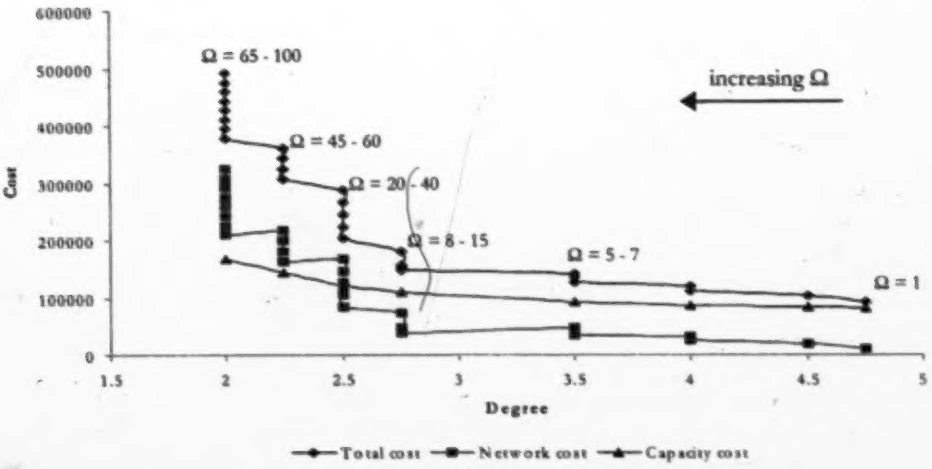


Figure 25 - Total cost of optimal MTRS solution vs. degree for network COST239-8n

Figure 25 shows a different view of the data. This is the plot of cost versus connectivity for varying Ω values. When we vary Ω, it has the effect of causing changes in the graph topology and hence the connectivity given constant cost functions and demand quantities.

We can see a cluster of data points (for example at graph connectivity of 2.5) when the effect of increased Ω value was not enough to cause a change in the optimal graph topology arising from the previously lower value of Ω. The effect then only lies in the increase in network costs. These points correspond to the plateau data points we saw in Figure 24. We can read Figure 25 from right to left (for increasing values of Ω) or from left to right (for decreasing values of Ω). We explain the plot for increasing values of Ω.

Starting with Ω = 1, the solution is a network of average nodal degree 4.75 having 19 edges (out of 28 possible edges). In Figure 25, we see that this solution represents the lowest cost network obtained over all values of Ω. At this value of Ω, capacity costs dominate the total network costs and the best-cost graph is one that has a high degree of connectivity to minimize the amount of capacity required for routing and restoration. The network graph is shown in Figure 26. As Ω increases, the network connectivity decreases. At Ω = 5, the optimal graph topology has a degree of 3.5 (shown in Figure 27). Figure 28 to Figure 31 shows the optimal network graph topologies for some increasing values of Ω up to Ω = 100. The actual number of edges included in the optimal solution for each value of Ω is listed in Table 8. It is important to emphasize the value of the design solutions shown in Figure 26 to Figure 31 because we know they are the optimal topologies for this test case, as they are the solutions to the MTRS problem solved to optimality on the universal edge set.

Table 8 presents the data for the network solution for the various values of Ω. As Ω increases, the network becomes sparser going from highly connected graphs (Figure 25) to low nodal degree graphs (Figure 31) requiring more capacity to be placed on each edge to satisfy the routing and survivability requirements. We see an increase in the mean capacity per span (or span utilization) in the network. The deviation in capacity placed on the edges also increases to a maximum value of 12 and then with the jump to a ring topology, falls back to a value of 2. This is in keeping with the structure of ring systems, where the utilization of the edges in the network is almost equal.

Table 8 – Edge Utilization

Ω	**Number of edges**	**Mean capacity per edge**	**Standard deviation of capacity per edge**	**Max. Cap per edge**	**Min. Cap per edge**
1	19	8.1	2.7	12	3
2	18	8.6	2.3	11	3
3	16	10.5	2.3	12	5
4	16	10.5	2.3	12	5
5	14	14.4	4.3	21	9
6	14	14.4	4.3	21	9
7	14	14.4	4.3	21	9
8	11	23.5	5.6	28	9
9	11	23.5	5.6	28	9
10	11	23.5	5.6	28	9
15	11	23.5	5.6	28	9
20	10	29.5	7.3	37	19
25	10	29.5	7.3	37	19
30	10	29.5	7.3	37	19
35	10	29.5	7.3	37	19
40	10	29.5	7.3	37	19
45	9	40.7	12	50	19
50	9	40.7	12	50	19
55	9	40.7	12	50	19
60	9	40.7	12	50	19
65	8	52.3	2	54	48
70	8	52.3	2	54	48
75	8	52.3	2	54	48
80	8	52.3	2	54	48
85	8	52.3	2	54	48
90	8	52.3	2	54	48
95	8	52.3	2	54	48
100	8	52.3	2	54	48

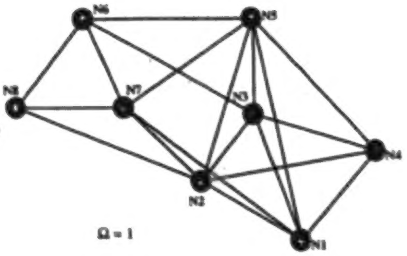

N6
N5
N8
N7
N3
N4
N2
Ω = 1
N1

Figure 26 – Optimal Solution for Ω = 1 for COST239-8n

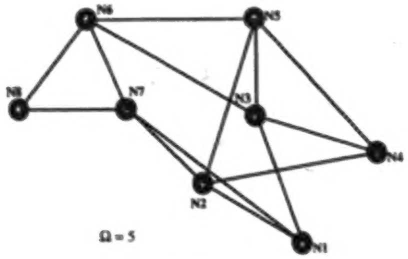

N6
N5
N8
N7
N3
N4
N2
Ω = 5
N1

Figure 27 – Optimal Solution for Ω = 5, 6, 7 for COST239-8n

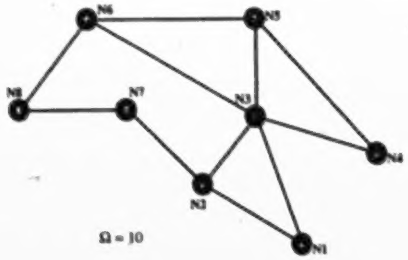

N6
N5
N8
N7
N3
N4
N2
Ω = 10
N1

Figure 28 – Optimal Solution for Ω = 8, 9, 10, 15 for COST239-8n

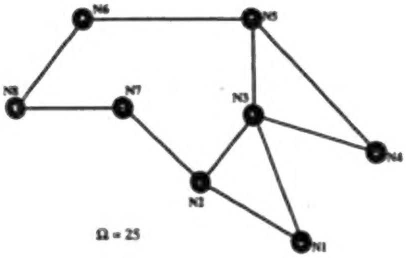

N6
N5
N8
N7
N3
N4
N2
Ω = 25
N1

Figure 29 – Optimal Solution for Ω = 20, 25, 30, 35, 40 for COST239-8n

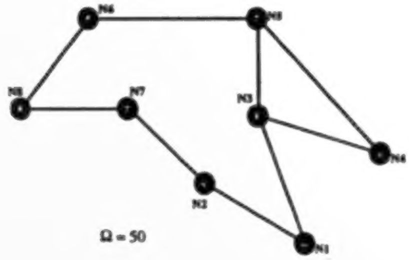

N6
N5
N8
N7
N3
N4
N2
Ω = 50
N1

Figure 30 – Optimal Solution for Ω = 45, 50, 55, 60 for COST239-8n

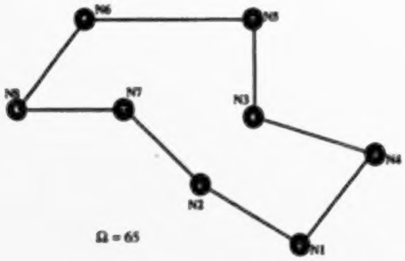

N6
N5
N8
N7
N3
N4
N2
Ω = 65
N1

Figure 31 – Optimal Solution for Ω = 65, 70, 75, 80, 85, 90, 95, 100 for COST239-8n

It is of great value when designing a heuristic to understand why the purely optimal MTRS solver chooses the edges that make up the solutions for the small test cases so that we can design algorithms that mimic the decisions for larger problems. At what cost relationship is one edge chosen over another? We observe that for some Ω values, the optimal network solution remains the same. And then for some Ω values, the best solution is one with an edge removed or added (corresponding to an increase or decrease in Ω respectively).

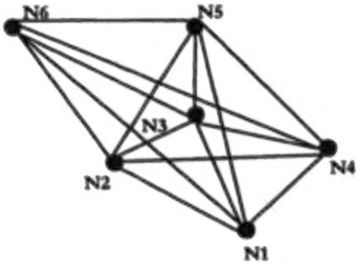
What causes the abrupt shift from the prior 'plateau' topologies when a jump finally occurs? What edge addition (or removal) breaks the balance? Looking closely at the test results, we cannot see any obvious correlating principle that determines the edge removed (or added) from one solution to another. We compare any two solutions in which the difference is the absence or removal of an edge from one to the other. We notice that the edge removed usually would have a low amount of capacity added to it (in comparison with other edges) but it is not always the edge with the least amount of capacity least loaded. Also, such an edge that is removed is associated with high edge establishment and capacity costs as determined by the length of the edge but it is always the longest edge. We can only conclude that this question can only be answered by optimization principles that simultaneously consider the effect of all the candidate edges on total network cost. Thus, a deterministic algorithm is not apparent that would be able to produce the optimal solution.

We now study the effect of changing the demand pattern on the network cost.

5.4 THE EFFECT OF DEMAND INTENSITY AND PATTERN ON NETWORK COSTS

An important aspect of the problem of determining the optimal network is the realization that the optimal network also depends on the demand pattern or the demand volume. What may be optimal for a given demand pattern may not be the same if the demand pattern is changed or if the volume of demand increases under the same Ω. By this we mean that even if we keep the sum total of demands constant but change the distribution of demands, the optimal solution could be based on a different graph topology. Or if we keep the demand distribution constant but increase the total volume of demands, the solution could still be a different network (tending towards greater connectivity as demand intensifies). In this section, we study the dependence of network costs on demand pattern and volume.

For this sub-study, we use the COST239-6n network shown below.



The demand patterns used were obtained as follows:

From the base demand pattern for COST239-6n network, we obtain new demand patterns C239 (x λ) where λ is a scaling factor for each demand pair in the base demand. For the C239-6n (x 2), C239 (x 5), C239 (x 10) demand patterns, λ = 2, 5, 10 respectively giving a family of four patterns which are identical in distribution but vary in total intensity. The use of these identical demand distributions was to study the effect of increased demand intensity on the optimal network topology.

'Random' is a uniform random model in which each O-D pair is assigned a demand quantity d_{ij} from a uniform random distribution of integers in the range $5 \le d_{ij} \le 30$ regardless of distance or nodal degree. The limits (5, 30) were chosen arbitrarily.

'GR50' is an inverse distance-weighted gravity model with demands generated with a mutual attraction product term weighted by the inverse of the distance between the nodes. This model is given by $k \times \deg(O) \times \deg(D) / L(O,D)$ where $\deg(O)$ is the degree of the demand origin, $\deg(D)$ is the degree of the demand destination and $L(O,D)$ is the distance (length of span) between the origin and destination nodes. We used the inverse distance weighted model because it is localized and can reflect the mutual demographic importance of the node pairs on the amount of demand intensity exchanged. The amount of demand exchanged between node pairs is inversely proportional to the distance between the nodes and we choose k to be 50. It follows that near or neighbor nodes would exchange more demand than with farther nodes. A visual appreciation of the variations in distribution of demand units between O-D pairs is shown in Figure 32. The C239 demand matrices are identical in distribution, varying only in absolute value therefore, only the C239 demand distribution is shown in the figure.

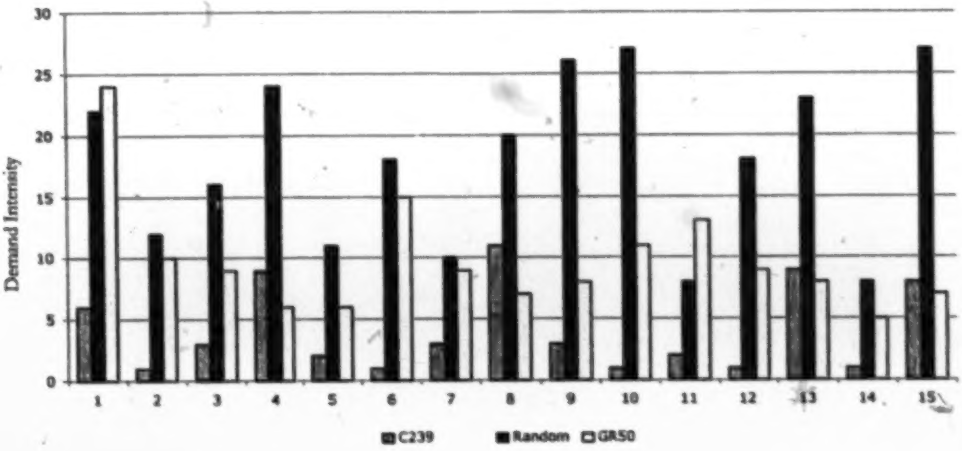


Figure 32 – Demand distribution between O-D pairs for three demand patterns

From Figure 32, we can see that the various demand patterns follow various flow distributions in the network. For example, D10 has a low value in C239 family of demand patterns, whereas, in the Random demand pattern, it has quite a large value. This would be taken into consideration in choosing the optimal network for each demand pattern. We can feel confident that the results we obtain are robust and are not solely applicable to a particular demand distribution. Table 9 lists the actual demand values.

Table 9 – Statistical properties of the demand patterns

Node pairs	**Demand**	**C239**	**C239 (x 2)**	**C239 (x 5)**	**C239 (x 10)**	**Random**	**GR50**
N1 - N2	D1	6	12	30	60	22	24
N1 – N3	D2	1	2	5	10	12	10
N1 – N4	D3	3	6	15	30	16	9
N1 – N5	D4	9	18	45	90	24	6
N1 – N6	D5	2	4	10	20	11	6
N2 – N3	D6	1	2	5	10	18	15
N2 – N4	D7	3	6	15	30	10	9
N2 – N5	D8	11	22	55	110	20	7
N2 – N6	D9	3	6	15	30	26	8
N3 – N4	D10	1	2	5	10	27	11
N3 – N5	D11	2	4	10	20	8	13
N3 – N6	D12	1	2	5	10	18	9
N4 – N5	D13	9	18	45	90	23	8
N4 – N6	D14	1	2	5	10	8	5
N5 – N6	D15	8	16	40	80	27	7
	Sum	61	122	305	610	270	147
	Mean	4.07	8.13	20.33	40.67	18	9.8
	Variance	11.67	46.65	291.56	1166.22	44	21.09
	Deviation	3.415	6.83	17.08	34.15	6.63	4.59
	Range	10	20	50	100	19	19

How do we analyze the demand distribution? We attempt to use numerical methods of description such as the mean, variance and deviation of the demand data as shown in Table 9 below for a 6-node network. The variability in the demand quantities for each demand pair is given by the variance and standard deviation values. Of all the statistical methods used to analyze the data (mean, variance, standard deviation, sum, range) as shown in the table, we feel that only the sum provides a stand-alone information by which we can describe the intensity of demand in the network as a whole. The standard deviation or the variance does not provide useful information when discussed separately from the mean. The range is a measure of the variability of the data but does not offer any information about the spatial distribution of the demand quantities. We elect to use the sum of demands (which we call the demand intensity) in our studies.

With the six different demand patterns listed in Table 9, we solve the MTRS problem for the COST239-6n network for varying values of edge costs (varying Ω). What we see as the resulting optimal topology for the different edge cost values supports the understanding that when capacity costs dominate (high demand intensities), the solution is much more connected than for lower demand intensity case for the same edge costs. The results are shown in the plots below.

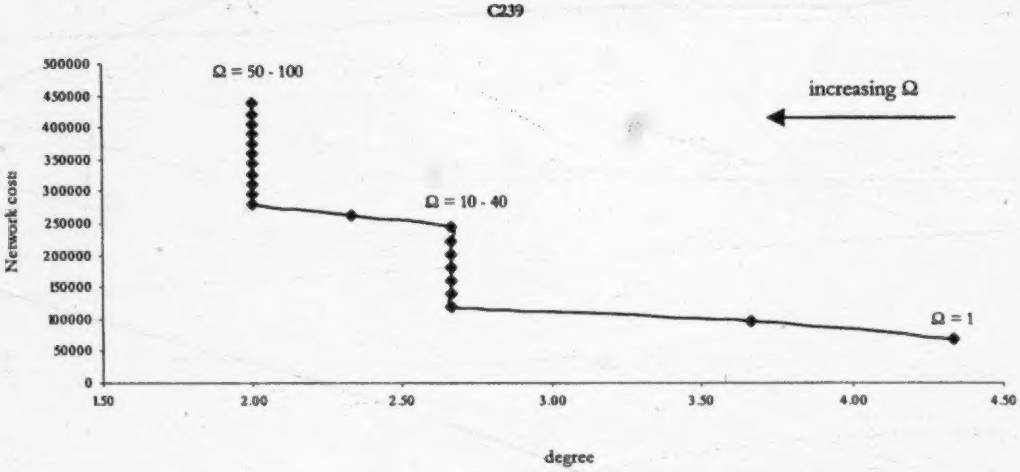


Figure 33 - Cost vs. degree for C239 demand pattern

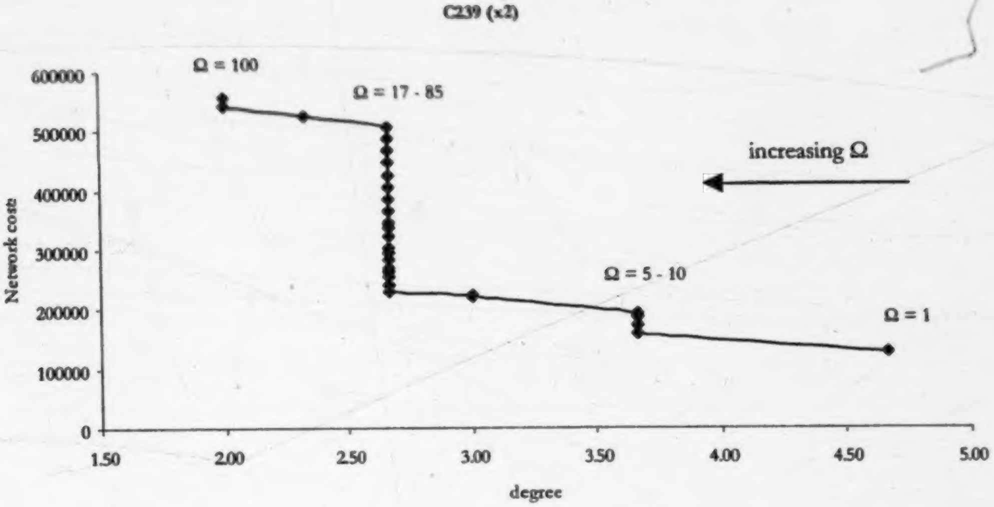


Figure 34 – Cost vs. degree for C239 (x 2) demand pattern

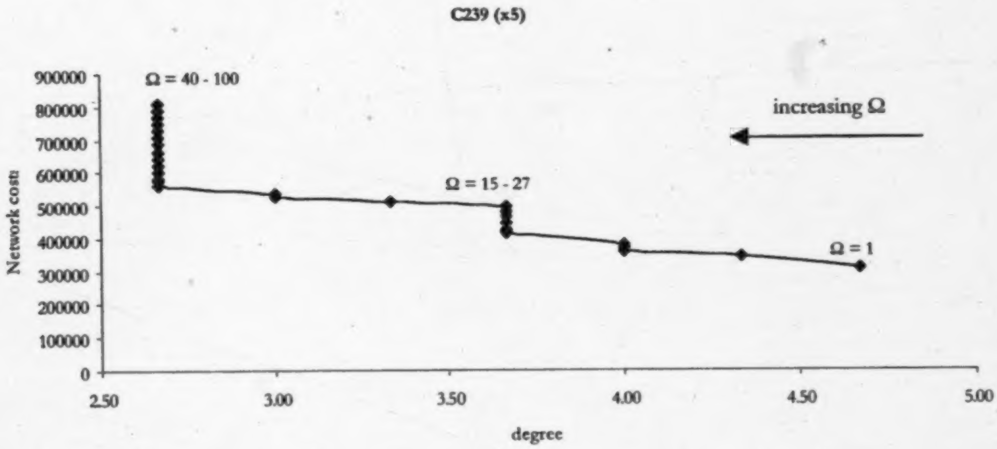


Figure 35 - Cost vs. degree for C239 (x 5) demand pattern

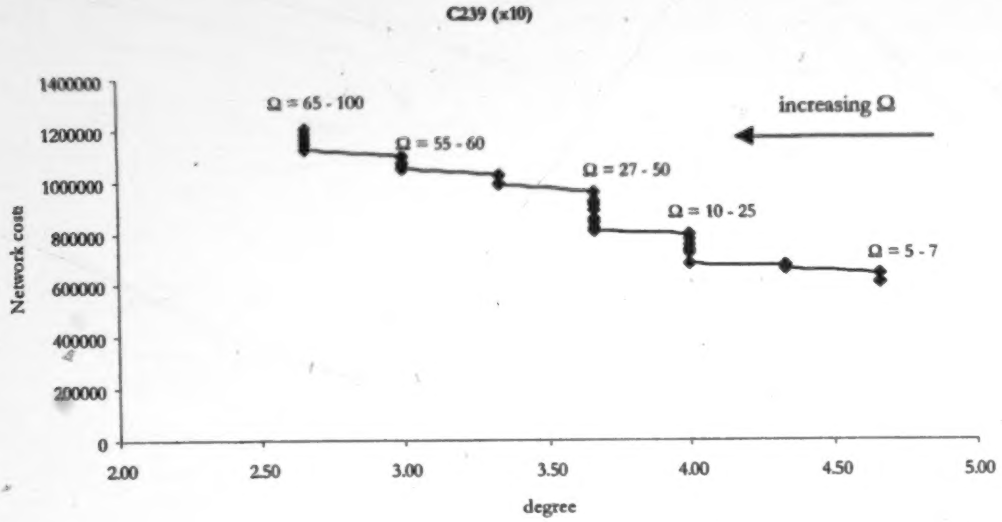


Figure 36 - Cost vs. degree for C239 (x 10) demand pattern

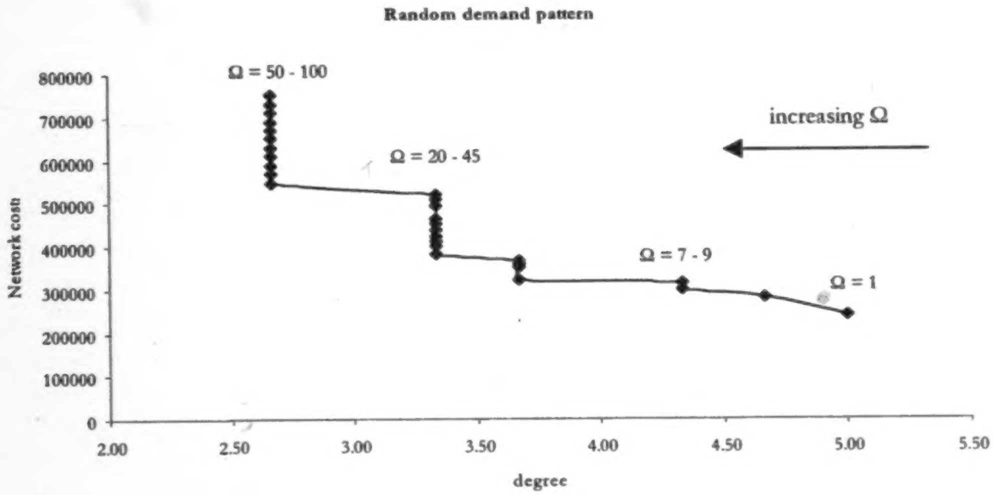


Figure 37 - Cost vs. degree for Random demand intensity

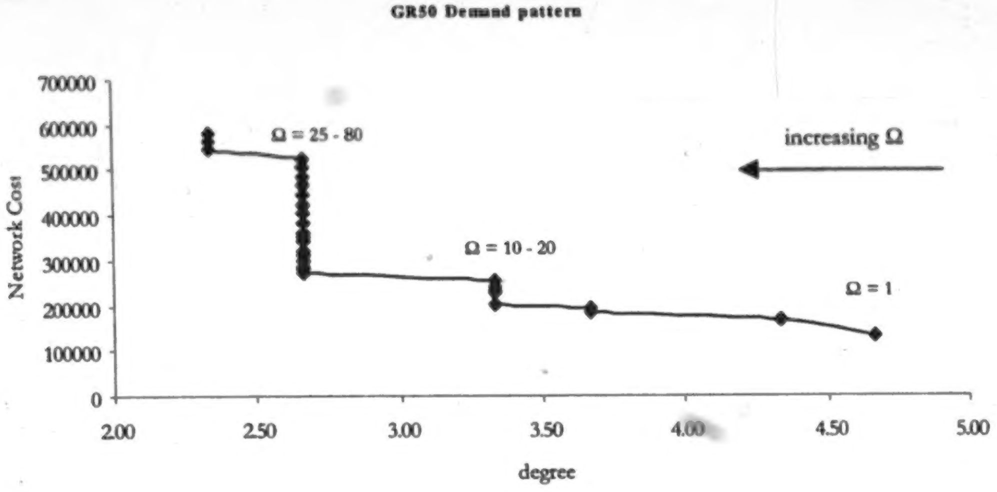


Figure 38 - Cost vs. degree for GR50 demand intensity

We define the point at which the edge costs and capacity costs for the network are equal before edge costs start dominating, which we call the 'cross-over point', Ω_c. Ω_c occurs at higher values of Ω depending on the demand intensity – the higher the sum of demand intensities, the higher the value of Ω_c. It follows intuitively that the higher the demand intensity, the higher the amount of capacity needed in the network (and its cost).

Figure 39 shows the resultant networks at selected Ω values for each of the basic demand patterns tested. Again, these are truly optimal solutions from a complete termination of CPLEX on the corresponding MTRS problem models with universal edge sets.

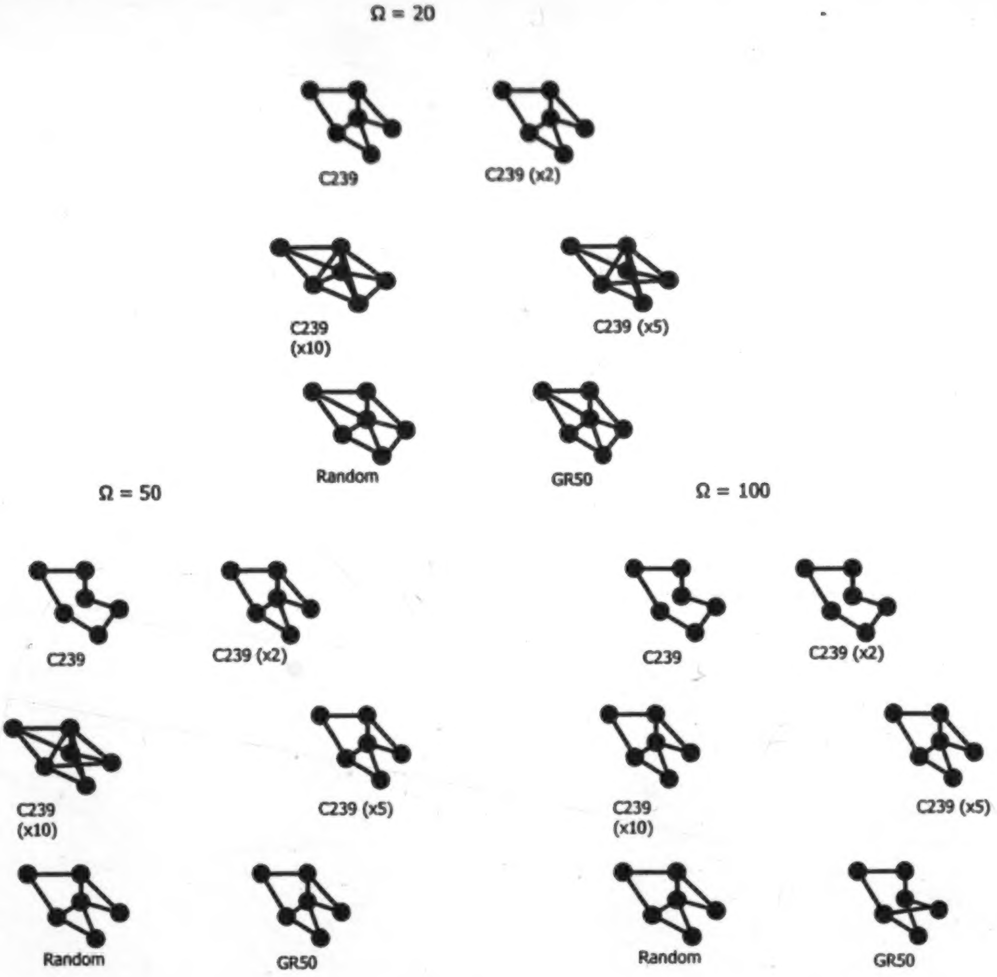

Ω = 20
C239
C239 (x2)
C239 (x10)
C239 (x5)
Random
GR50
Ω = 50
C239
C239 (x2)
C239 (x10)
C239 (x5)
Random
GR50
Ω = 100
C239
C239 (x2)
C239 (x10)
C239 (x5)
Random
GR50

Figure 39 – Optimal solutions for Ω = 20, 50, 100 with varying demand patterns

When we look at the optimal networks selected for the different demand patterns for the same edge and capacity costs as shown in Figure 39 above, we are in a better position to appreciate the effect the demand (pattern or volume) has on the solution. In Figure 39, at Ω = 50, the solution of C239 is now a ring system while the corresponding solution for C239 (x10) is still highly connected. This is because when the demand intensity is high, the network requires that a high amount of working and spare capacities be placed on the spans. Under this condition, including more edges in a network would be appropriate because it would reduce the total amount of capacity placed in the network. Other network solutions at this value of Ω are still reasonably mesh-like.

At Ω = 100, solutions for C239 and C239 (x2) are rings while other networks have a nodal degree of 2.67 (the GR50 solution has a degree of 2.33). The solution for C239 at Ω = 100 is expected because at Ω = 50, the solution was already a ring. The ring system is the minimal edge bi-connected network that can satisfy the survivability requirements for a transport network. Once a network has a ring structure as its optimal network at a given value of Ω and D, an increase in Ω can only increase the total network costs but cannot cause an increase the connectivity of the topology.

Recall that the objective function for the MTRS optimization is $\min \sum_{ij \in E} c_{ij}.\left(w_{ij} + s_{ij}\right) + F_{ij}.\delta_{ij}$ and $\Omega = F_{ij}/c_{ij}$. We want to know the topological implications of increasing any of the parameters of this function. Scaling the capacity cost c_{ij} by some multiplicative factor λ has the same effect as scaling the demand quantities by λ (the latter causes a doubling of w_{ij} and s_{ij} values) as given by the expression

$$2c_{ij}.\left(w_{ij} + s_{ij}\right) + F_{ij}.\delta_{ij} = c_{ij}.2\left(w_{ij} + s_{ij}\right) + F_{ij}.\delta_{ij}$$

where λ = 2. Again, scaling the capacity cost c_{ij} by λ has the same effect as scaling Ω by $1/\lambda$ and is the same as scaling the edge costs by $1/\lambda$. When we double the demand quantities served, we have the same effect (same topology) as when we double the capacity costs. If we also double the fixed costs (with a doubling of the capacity costs or demand quantities), we essentially cancel out the effects of any increase leaving the same solutions.

In conclusion, we have seen the following to be true of the effect of edge-to-capacity cost ratio Ω and demand intensity (D) on network costs:

1. As Ω increases (increasing edge costs with other factors being the same), the optimal solutions become sparser (includes fewer edges).
2. As the D increases with other factors being the same, the optimal solutions become more connected (includes more edges). Note that $D = \sum_{r=1}^{D} d^r$ where r indexes over the set of demands D and d^r is the demand quantity for the r^{th} O-D pair.
3. As D increases, the network cost eventually becomes dominated by capacity costs over edge costs. An example of this is shown by the C239 (x 10)-demand pattern (Figure 36).
4. When D increases at the same rate with Ω, there is a stable architecture which remain optimal over increments and variations of D and Ω.

We have seen how the value of Ω affects the optimal topology design. A timely question then is: For research studies, should we insulate the value of Ω from the demand intensity by making it a constant or should it be allowed to vary with the intensity? In reality, we understand that the value of Ω is and should be independent of D. The cost of acquiring the right-of-way, or laying the fiber cable is independent of the amount of traffic that link or span would carry.

As DWDM technology advances, capacity costs (per bit/sec of fully loaded systems) decrease. However labor and material costs (which make up the bulk for the fixed costs) are continuously on the rise. The optimal topology design thus is expected to shift towards sparseness in practice. But as demands continue to increase, the topology design shifts towards a more connected topology. These three factors pull and push the optimal topology in both directions and the optimal design is one that has a good balance of the three factors that affect the network design.

One purpose of the study of the effect of Ω and D on network cost was to determine the appropriate value of Ω to use in further studies, bearing in mind that the value of Ω affects the solution we obtain. For the set of demands that we use in this research, it is meaningful to choose a value of Ω at which both capacity and edge costs are active in the solution. Based on the test results, we choose a value $\Omega = 25$ (which is about the average value of Ω_c we see with some other test networks and demand patterns).

By choosing $\Omega \approx \Omega_c$ on this basis for further research studies, we know that we are studying the phenomena and problem of topology design in the region where both edge decisions and routing/capacity decisions are essential to an optimal network design. We also know from experience in prior work [GrDo01] that the observed computational complexity of the MIP solution to MTRS is at its worst when both sets of decision variables are almost equally weighted in terms of their contributions to the objective value.

6. LOCAL SEARCH HEURISTICS: IMPROVING ON THE GRAPH SIFTER

6.1 INTRODUCTION

We have identified and developed the concept of the graph sifter: generate network graphs and pass qualified (or repaired) graphs to a solver for design cost evaluation. We repeat this process until some set stopping criteria has been satisfied. Examples of the stopping criteria include some time limit or when the rate of decrease in the costs of the network designs found crosses some threshold. However as so far described, this is still a random search through the solution space. Therefore, some strategy can still be developed for a more targeted and directed search. This leads us to develop the following two local search heuristic approaches called the Sweep-Search (S-S) and Iterative Sampling (I-S) heuristics.

Local search heuristics apply to the general problem of minimizing a function $f(x)$ over a *finite* set F of (feasible) solutions. A local search heuristic enumerates solutions within the neighborhood $N(x)$ of a solution x, where $N(x)$ is a (small) subset of F. A successor $\bar{x}$ is then chosen from $N(x)$ according to some rule. This process is repeated with $\bar{x}$ replacing x until some stopping criterion is met. [Bert98]. Thus a local search method is characterized by:

a. The method for choosing the starting solution, x

b. The definition of the neighborhood $N(x)$ of a solution x

c. The rule for selecting a successor solution from within $N(x)$

d. The termination criterion

In Chapter 3.2, we discussed the issues with solving the complete MTRS problem. We then concluded that one approach would be to limit the solution space (number of edges under consideration) before employing the MIP solver to evaluate the routing and capacity implications. The next section describes a heuristic for a priori limitation of the candidate edge set, thus allowing our Sweep-Search and Iterative Sampling heuristics to converge faster to a near-optimal solution.

6.2 A PRIORI LIMITATION OF THE CANDIDATE EDGE SET

When one looks at the graph of a network that is the optimal solution to an MTRS problem (where such can be obtained), there are certain properties of the network that are often overlooked. Yet, these properties are partly what make the solution optimal and are also observed in real network examples. Diagrams of real world networks also exhibit some of these properties. Chiefly among these, one important property of interest is the relationship between the nodes that are directly connected (adjacent). Why were some edges included in a solution selected and not others? We may not have easy answers to this question since the answer ultimately involves the historic, geographic and demographic details. However, we can make certain strong empirical observations about the edges that are included in a graph.

In generating random graphs given the universal edge set, the visual representation of most of the graphs would surprise any transport network provider. They often include edges that go from one end of the plane to another, edges that are not found in any real or feasible transport network. Most of the topologies generated would not be considered plausible. If the edge set is limited to contain specific choices, we can then generate graphs that are visually appealing as feasible choices.

In the complete MTRS problem, we do not have specific choices, rather we want to consider all edges. However we also know that if we can reduce the edge set under consideration from the complete edge set of size s to one with m edges, we reduce the problem size by the magnitude of $s-m$. Note that each edge admitted doubles the problem size ($2^{(m+1)} = 2^m \cdot 2$ possible graphs to consider).

How then can we reasonably achieve this? We study more closely what sorts of edges are chosen in solutions and therefore, eliminate candidates that are less likely to be included in the optimal solution. A simple observation is that in all the real transport networks studied, no node connects directly to neighbors at great distances away. In fact, transport networks are known for tending to connect between next-door neighbors (spatially) and not across the plane arbitrarily. Edmonton is much more likely to have a direct physical layer connection to Calgary, Vancouver and Saskatoon than to New Orleans. Based on this simple observation, we can very conservatively eliminate certain edges that would not be potential choices (for example, edges that represent a reach across the entire network diameter).

6.2.1 A PRIORI LIMITATION OF EDGE SET – REDUCTION METHOD 1

More specifically, if we want to consider only graphs of $d_{\max} \leq 5$, then we can prune down the candidate edge set by allowing each node to keep only the edges that connect it directly to its five nearest neighbors (where "near" means in Euclidean space). By this means, we are able to reduce a problem of size 11 nodes and 55 candidate edges for example, to a problem of 11 nodes and 23 candidate edges, which is definitely easier to solve. The pseudo-code for this algorithm is:

```
For all nodes, do
    Select $d_{max}$ incident edges to spatially nearest neighbors
    Remove all other incident edges from the candidate edge set
```

Table 10 shows the magnitude of reduction of solution space with this method of elimination.

Table 10 – Magnitude of reduction for the edge set (for $d_{\max} = 5$)

$\lvert N \rvert$	$E = \lvert N(N-1)/2 \rvert$ **(Universal Edge Set)**	$E_{reduced}$	**Reduction ratio**
11	55	23	2^{32}
10	45	21	2^{24}
9	36	19	2^{17}
8	28	18	2^{10}
7	21	17	2^{4}
6	15	15	-

Of course, with problem sizes of 6 nodes and $d_{\max} = 5$, we achieve no reduction because $d_{\max}$ is already 5. However, we concede that solving the MTRS problem on problem instances of this size or smaller is not a problem and is trivial.

The next issue is how the solution quality may be affected by this reduction in problem size. Is there any realistic concern that we could be eliminating candidate edges that may have potentially been selected? Or are we as intended, weeding out edge candidates which would never have been chosen anyway in MTRS and therefore, limiting the solution space to consider, speeding up execution times?

We try to validate this heuristic. Since it is difficult to obtain optimal solutions for problem instances of 10 nodes and larger, we use problem sizes of 7 nodes up to 9 nodes, for which optimal solutions can be obtained. The validation involves running the MTRS with the edge set populated by all possible edges, again solving the reduced instance of the MTRS problem with the edge set reduced by the distance-based elimination principle already discussed. The 3-step heuristic outlined in Chapter 2 was carried out on the networks (both the complete and reduced MTRS problems) for comparison of performance (cost of solution with execution time).

Applying the edge-limiting heuristic, we saw that for the problem instances for which we are able to completely solve the problem, the optimal solution of the complete MTRS problem is exactly the same solution as the reduced MTRS problem! Thus we have been able to achieve a reduction of the problem space without compromising the quality of our solution and obtained the obvious benefit is the reduction in computation time. The table below summarizes the results we obtained.

Table 11 – Edge limiting solutions (MTRS)

$\|N\|$	**Complete MTRS (All edges)**			**Reduced MTRS (Reduced edge set)**		
	Solution	**Network cost**	**Time (sec)**	**Solution**	**Network cost**	**Time (sec)**
7	191358	103850	47	191358	103850	9
8	224389	104850	5721	224389	104850	28.74
9	299673	143775	333451	299673	143775	84
10	N/A	N/A	-	372723	152350	92
11	N/A	N/A	-	430347	200600	~800

Table 12 shows the results for the prior 3-step MTRS heuristic (from [GrDo01] and summarized in Chapter 1.3) solved on the distance-limited edge set. The results seemed to follow the same trend (the same solution whether the edge set was the complete universe or the reduced version) until the 11-node problem instance. We notice that in this case, the solution with the complete edge set is lower in cost that the solution with the reduced edge set. Looking at the optimal topology found by both solutions, we notice that the solution for the complete problem made use of an edge that had indeed been eliminated by the a priori reduction algorithm. This indicates that the solution quality had been compromised.

Table 12 – Solution after edge limiting heuristic

$\|N\|$	3-Step MTRS Heuristic (All edges)			3-Step MTRS Heuristic (Reduced edge set)		
	Solution	Network cost	Time (sec)	Solution	Network cost	Time (sec)
7	191358	103850	4	191358	103850	4
8	224389	104850	7	224389	104850	5
9	304383	128350	19	304383	128350	6
10	378567	158100	153	378567	158100	15
11	418057	186850	602	430347	200600	36

An interesting question would be whether an optimal solution of the complete MTRS problem (with the universal edge set) for the 11-node network would make use of the edges eliminated by this method. However, 10 days of running the solver for this problem (on the universal edge set) did not yield an optimal or improved solution for comparison.

6.2.2 IMPROVING A PRIORI LIMITATION METHOD – REDUCTION METHOD 2

An issue we noticed is that this method of elimination of edges could result in a candidate set of edges giving $\bar{d} < 5$ when the complete topology had a value of $\bar{d} > 5$ unless it had exactly $\bar{d} = 5$. The reason this is possible is demonstrated using the following example for a network with node size > 6:

Let $d_{\max} = 5$. Starting with node 1, we select the 5 shortest edges. These include edge (1,4) – the edge between node 1 and node 4. When we get to node 4, edge (1,4) however is not among its 5 closest neighbors. This results in the edge being dropped from further consideration, leaving node 1 with 4 possible neighbors and node 4 with 5 possible neighbors. We do this process for all nodes. This leaves a network of degree $\bar{d} < 5$ when we could have had it as $\bar{d} = 5$.

The edges left in the solution may depend on the order in which the nodes are considered. Using the example above, suppose node 4 is treated before node 1, edge (1, 4) would still be eliminated. However, the next longest edge for node 1 (which was eliminated when node 1 was considered first) would be admitted to the solution. This would leave both nodes 1 and 4 with 5 possible adjacent spans.

This understanding of what can happen leads directly to alternate selection criteria with the following pseudo-code:

```
For all nodes, do
    Select $d_{max}$ shortest adjacent spans
    For other adjacent spans, starting with the most expensive (longest)
            Remove only if the adjacent node can connect directly to at least $d_{max}$ other neighbors
```

Using the added criteria allows many more edges to be admitted for consideration. For example, the edge set allowed for the 11-node problem instance grows from 23 edges after reduction method 1 to 32 edges after reduction method 2 and as expected, the execution time increases.

6.2.3 IMPROVING A PRIORI LIMITATION METHOD – REDUCTION METHOD 3

Another issue with reduction method 2 is that it does not take into account the fact that in many networks some nodes act as natural hubs and have a higher nodal degree than other nodes (see in real transport network graphs and in the optimal solutions we obtain for the MTRS problem). Usually, these nodes have many short incident edges. This leads to a further alternate means for restricting the edge set considering the set of all edges instead of a node-by-node approach. This is to sort the candidate edge set in ascending order of cost (length). We then determine the number of edges m that gives a network of the desired average connectivity $\bar{d}$ and remove edges longer than the m^{th} edge (starting with the longest) as long as its end nodes each have a degree greater than some number k. The pseudo code now is:

```
Sort the candidate edge set in ascending order based on length of edge
Select $m$ cheapest (shortest) spans
For all other spans, starting with the most expensive (longest)
        Remove if adjacent nodes have at least $k$ other neighbors
```

With this, we achieve a reduction of the overall candidate edge set, without imposing nodal limits on degree. Nodes with short edges are free to keep all their edges. This method depends on two variables, m and k. Let us consider the effect of k.

What is an appropriate value for k? What is the probability that in a real network or an optimal network solution, a node having very expensive (long) edges would have a degree greater than 2 (nodal degree of 2 is the minimum required to achieve survivability)? The solutions we see show such sparse connections for nodes with all long edges and the nodes with short edges are well connected. This is due to the cost associated with admitting long edges into the solution – increased fixed costs and capacity costs that are functions of the length of the edge. Also, high degree nodes tend to have many close neighbors in the plane.

In Chapter 5, we saw the dependence of the optimal solution on the edge-to-capacity cost ratio, Ω. This suggests that the value of Ω should affect or determine the choice of edges that are permitted for consideration. When Ω is low, the possible edge set under consideration would be more populated than for increasing values of Ω. This leads to the idea of an adaptive k that depends on the value of Ω. We create equivalence classes of Ω, similar to what we saw from the optimal solutions for which the network remained the same for some values of Ω. For the case of two equivalence classes, we define a boundary parameter α such that:

$$k = \begin{cases} 4, if\,\Omega < \alpha \\ 3, if\,\Omega \geq \alpha \end{cases}$$

We choose the values $k = \{3,4\}$ because we want a candidate edge set with $d_{\max} = 5$. Having allowed nodes with shorter edges (which would act as hubs or natural transit nodes), allowing nodes with long edges to be part of the solution space causes an increase in execution time, which does not pay off with increased solution quality. Limiting the nodal degree for the nodes with long edges helps keep only potentially good candidates and keeps the nodal degree of the network to a value of $d_{\max} \leq 5$.

The second parameter is m. We use the relationship $m = d_{\max} \cdot |N|/2$ to obtain our value of m with $d_{\max} = 5$. However, we find that the edge set we allow exceeds our maximum desirable nodal degree. For the 8-node and 28-span problem, we can reduce it to a problem size of 8 nodes and 15 spans, with a value of $k = 3$. For $k = 4$, we reduce this problem instance to 8 nodes and 21 spans. For $\Omega = 25$, the optimal solution has 10 edges. We see that our solution space includes more edges than we would consider optimal for this value of Ω. We also find that $\lim_{n \to 0} m \to N(N-1)/2$, i.e. the whole universal edge set.

6.2.4 IMPROVING A PRIORI LIMITATION METHOD – REDUCTION METHOD 4

A further modification to this method would be to change how we select m. We conducted some tests to find an appropriate value for m. For each test network, we rank the edges in order of ascending length. Then from the optimal solution to the MTRS problem for the test case, we calculate the proportion of short edges that were included in the solution. We saw that at most, 0.4 of the edges (ranked as outlined) are included consecutively starting from the shortest, then other longer edges are included. We then determine that $m = 0.4N(N-1)/2$ is characteristic and set $k = 3$. We recognize that some nodes/vertices in a transport network acts a hubs and therefore would have degree greater than 3. We also believe these nodes would connect to near neighbors and therefore, all spans which they would use are accounted for by the initial m spans selected. We then have an initial graph with $\bar{d} \geq 0.4(N-1)$.

How does this new method compare with the others? It is very effective in that all edges that could be (and are) included in any optimal solution are admitted for consideration. The magnitude of reduction of the problem space we see is shown in Table 13.

Table 13 – Reduction for $k = 3$ and $m = 0.2N(N-1)$

$\lvert N \rvert$	$E = \lvert N(N-1)/2 \rvert$	$E_{reduced}$	**Reduction ratio**
11	55	25	2^{30}
10	45	21	2^{24}
9	36	17	2^{19}
8	28	15	2^{13}
7	21	12	2^{9}
6	15	10	2^{5}

The parameter choices of $m = 0.2N(N-1)$ and $k = 3$ have proven to give good solutions, with the edge set consisting of all optimal edge candidates on all our tests of networks with node sizes up to eleven. We applied it to a twenty-node network. The candidate edge set was reduced from 190 possible edges to a candidate edge space of 76 edges (degree 7.6) while a time-limited run of MTRS optimization on the same network yielded a network of degree 2.4 (24 edges). So clearly, the candidate edge set is still well over populated even after limiting the edge set.

6.2.5 IMPROVING A PRIORI LIMITATION METHOD – REDUCTION METHOD 5

The consideration of the expected average nodal degree of the solution drives a new choice of the parameters *m* and *k* for the limiting heuristic. We want to include all the potentially good edge choices in the problem, yet keep the edge size as close to optimal as possible. We walk a fine line as we try to strike a balance between these two objectives. This yet leads to another choice of the parameters *m* and *k*. We consider an adaptive k that depends on the problem size ($|N|$). We create two equivalence classes of $|N|$ and the boundary parameter α such that:

$$k = \begin{Bmatrix} 3, if\ |N| < \alpha \\ 4, if\ |N| \geq \alpha \end{Bmatrix}, \alpha = 11$$

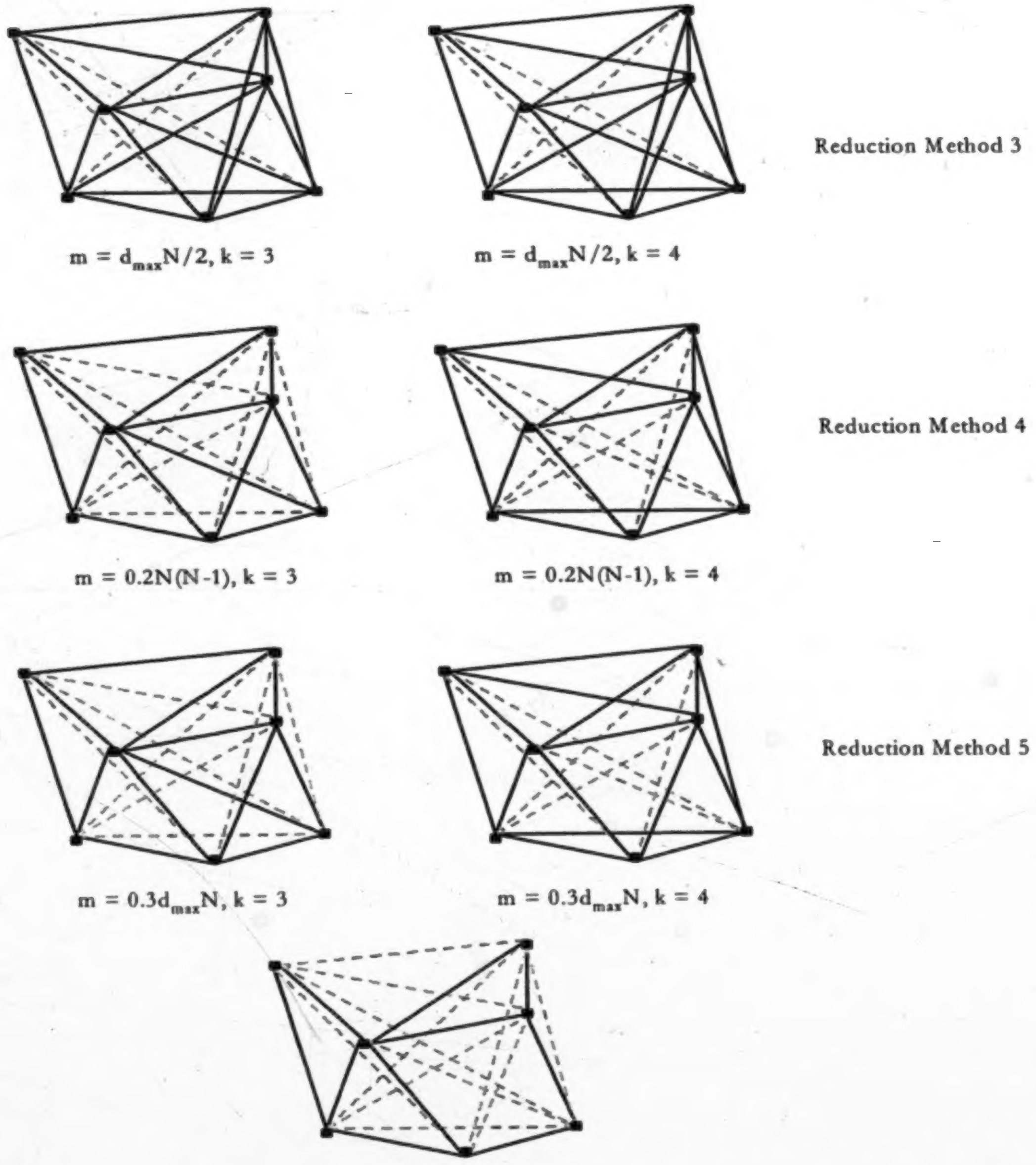
Again, we choose the value $k = \{3,4\}$ because we want a candidate edge set representing $d_{\max} = 5$. We empirically set $\alpha = 11$ because our experience with the heuristic is that for networks with $|N| < 11$, we obtain a good or optimal solution with $k = 3$. For $|N| \geq 11$, we allow proportionally more edges into the solution to improve our chances of the optimal solution being contained in the edge set. We set the choice of m to be $m = 0.6 \times d_{\max}.|N|/2$. This allows 60% of the total number of edges that a network of $d_{\max}$ would have, to be first selected from the cheapest (shortest) edges and these edges typically make up the optimal solution. The choice of *m* is a blend of the two previous values of *m* ($m = 0.2N(N-1)$ and $m = d_{\max}.|N|/2$). Table 14 shows the effect on networks of varying sizes. The effect on the smaller network problems is essentially the same except that the 11-node network has an extra edge included while the 10-node network is reduced by two edges. The 20-node problem now has 45 edges ($d_{\max} = 4.5$ vs. initial 7.6).

Table 14 - Reduction for $k = \{3,4\}$ and $m = 0.3 \times |N|.d_{\max}$

$\lvert N \rvert$	$E = \lvert N(N-1)/2 \rvert$ **(Universal edge set)**	$E_{reduced}$	**Reduction ratio**
26	325	59	2^{266}
23	253	51	2^{201}
20	190	45	2^{145}
15	105	34	2^{71}
11	55	26	2^{30}
10	45	19	2^{24}
9	36	17	2^{19}
8	28	15	2^{13}
7	21	12	2^{9}
6	15	10	2^{5}

In the tests carried out (where optimal reference solutions are available), the optimal topology has its edge set as a complete subset of the edges left after the limiting procedure. The edge limiting procedure is a necessary part of our ability to provide a good and fast solution to the MTRS problem at non-trivial sizes. Therefore, we use the edge limiting heuristic as the first step before employing the topology search heuristics.

Figure 40 shows the reduced edge set using the reduction methods 3, 4 and 5 with $k = \{3,4\}$ for 7n21s1 network including the optimal network topology obtained from the MTRS solution given the complete edge set.


Reduction Method 3
Reduction Method 4
Reduction Method 5

$m = d_{max}N/2, k = 3$

$m = d_{max}N/2, k = 4$

$m = 0.2N(N-1), k = 3$

$m = 0.2N(N-1), k = 4$

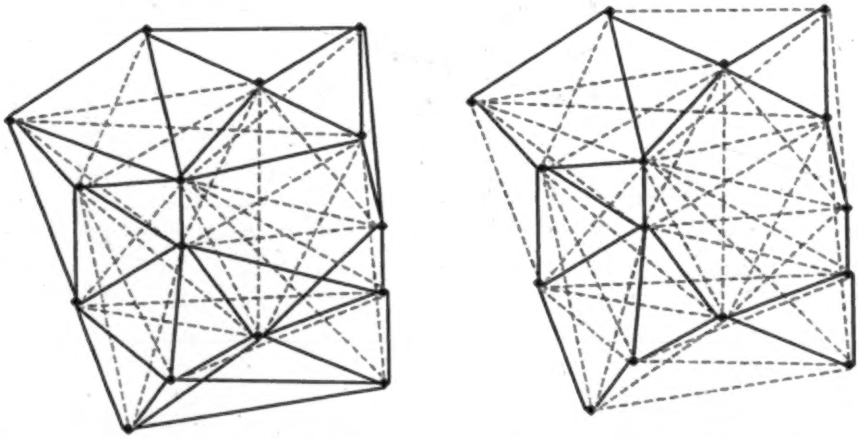
$m = 0.3d_{max}N, k = 3$

$m = 0.3d_{max}N, k = 4$

Optimal network topology

Figure 40 - Edge limitation with various *(m, k)* parameters for network 7n21s2

Reduction method 4 and 5 produce similar reduced edge sets for small size networks. As the network size increases, reduction method 5 produces a smaller edge set than reduction method 4. Figure 41 shows the reduction methods 4 and 5 for the 15-node, 59-spam network. The reduction method 4 did not reduce the edge set but reduction method 5 reduced the edge set from 59 to 34 candidate edges. The candidate edges are indicated in solid lines. We see also the topology for the best cost network found by the Iterative Sampling heuristic. The topology is sparse (degree of 3.07) with 23 edges.



Reduction method 4 – all 59 edges
Reduction method 5 – 34 edges

Iterative Sampling solution – 23 edges

Figure 41 – Reduction methods 4 and 5 on a larger network (15n59s)

In further studies, we use the a priori edge limitation strategy of reduction method 5 with the parameters $k = 3$ and $m = 0.3 \times |N|.d_{\max}$.

In the introduction of our approach in this work, we outlined two strategies we will employ:

1. Limit the search space
2. Search only over lightweight topology space

Having developed the heuristic for limiting the search space (strategy 1), we now present two heuristics we developed for searching over lightweight topology space (strategy 2). We start with the Sweep-Search heuristic presented in the next section.

6.3 SWEEP SEARCH HEURISTIC

The main idea of this heuristic is to systematically (rather than randomly) search the topology range within the set minimum (d_{min}) and maximum (d_{max}) nodal degree of feasible and plausibly optimal transport graphs. Using the nodal degree and cost information derived from the ongoing search, we adaptively narrow the search bracket to search even more closely over the region within which we believe the optimal graph topology may lie for the particular network in question. The Sweep Search idea exploits the fact that we know from first principles and preliminary studies that there is a nearly continuous curve of total costs versus connectivity with an essentially vague minimum basin as in Figure 8.

For studies and tests here, d_{max} and d_{min} are set as 5 and 2 respectively. Having d_{min} of value 2 will always be the case but $d_{max} > 5$ is possible if one was concerned that a very high demand or low Ω situation could put the optimal graph connectivity higher than 5. We start from a single graph instance of a highly connected network of nodal degree d_{max}, which we pass to a solver only for capacity design (routing and working plus spare capacity costs) using either JCP or SCP span-restorable mesh capacity design models described earlier in Chapter 2.3. This is an example of a situation where a graph generated by any random generator would be repaired instead of cycling through many graph generations to find one that satisfies this requirement. In our studies, we use the JCP – Node Arc capacity design method.

After the routing/capacity design has been carried out for the graph and the total network costs is known, we *remove* an edge to change the connectivity of the graph and hence the cost of edges in the network. This new graph is passed to the solver for capacity design. The edge removed may also change the cost of provisioning capacity for the network graph.

The process is repeated until no edge can be removed from the graph without violating bi-connectivity constraints or the average nodal degree of the graph is $d_{\min}$. The process reverses here and starts all over with edge *additions* replacing edge removals. To the sparse graph we now have, we add an edge, solve the graph to the solver for routing and capacity design. We continue to do this until the graph have an average nodal degree of $d_{\max}$. Thus, we never lose bi-connectivity but are always evolving the actual graphs being considered, walking them back and forth through the region where the optimum is thought to lie.

When we walk from a highly connected graph to a sparse graph (or a graph with connectivity $\geq d_{\min}$), we call it 'pare-down'. Similarly, walking from a sparse graph and adding spans (up to connectivity $\leq d_{\max}$) is to 'pare-up'. A pare-down plus a pare-up sequence is called a 'sweep'. This heuristic concept is described by Figure 42 and shows the expected behavior of the heuristic search.

For each sweep, the nodal degree ($\overline{d}$) of the network graph that produces the lowest cost and its cost are noted. After a number of sweeps, new nodal degree limits are set to be the 'reduced $d_{\max}$' and 'increased $d_{\min}$'. The new limits are expected to be within the region of minimum cost network degrees obtained over the sweep period as shown in Figure 42 above. From the figure, we could expect that these new limits would be some factor of the average nodal degree $\overline{d}^{avg}$ over all minimum cost networks obtained in each sweep ($\overline{d}^{avg} = \sum \overline{d}$ /number of sweeps).

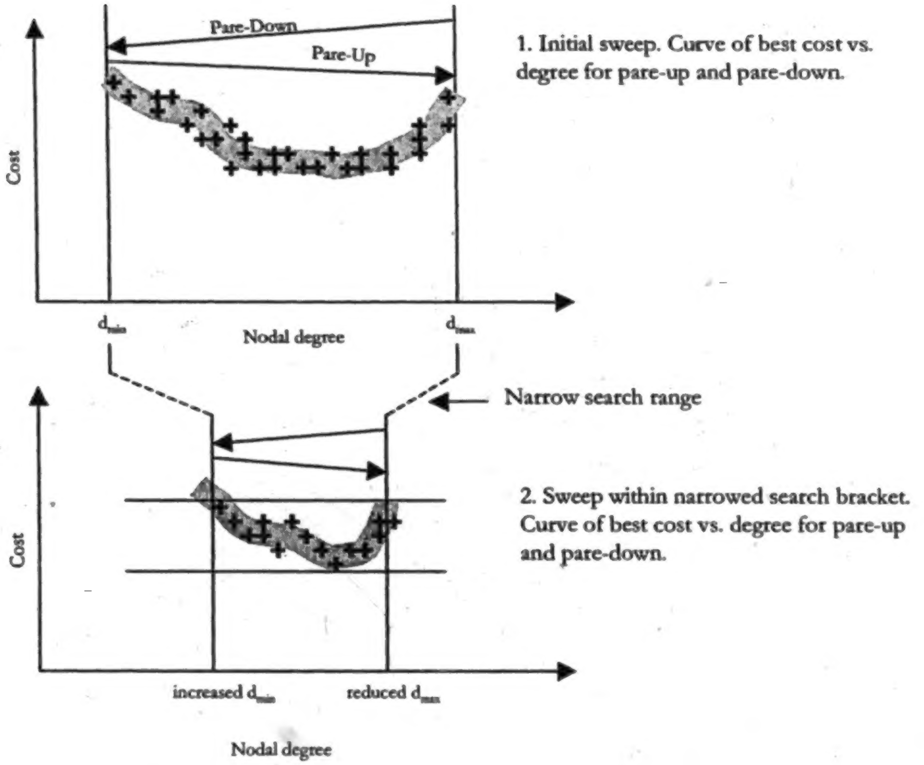

Pare-Down
Pare-Up
1. Initial sweep. Curve of best cost vs. degree for pare-up and pare-down.
Cost
d_min
Nodal degree
d_max
Narrow search range
2. Sweep within narrowed search bracket. Curve of best cost vs. degree for pare-up and pare-down.
Cost
increased d_min
reduced d_max
Nodal degree

Figure 42 - Oscillating Sweep Search strategy on topology

So the next question to ask is: what factor of $\overline{d}^{avg}$ should we use? Should the reduced d_{max} be $\overline{d}^{avg}$? Some lower or higher value? The increased d_{min} too? We know that $\overline{d}^{avg}$ by virtue of its being average, is higher than the absolute minimum cost network so far found. We set the reduced d_{max} to a value of 1.2 x $\overline{d}^{avg}$ and the increased d_{min} to 0.8 x $\overline{d}^{avg}$. In many cases, we found that the reduced d_{max} was still very high compared to what the optimal solution would turn out to be. We were spending time searching through topologies that had higher connectivity than what the optimal solution would likely be.

We also found that the increased d_{min} may limit the solution space into more highly connected regions than the optimal solution. Since we have restricted our search to topologies having a nodal degree between the reduced d_{max} and increased d_{min}, in some cases, we have cut-off the solution space within which lies the optimal solution. Sometimes, the calculated value for the increased d_{min} would be less than 2, which we know is not feasible for a bi-connected network. We believe we can assume that the maximum degree of the optimal topology will always be less than or equal to $\overline{d}^{max}$ but may be in the region of $\overline{d}^{avg}$ and we know that the minimum degree will always be greater than or equal to 2.

We have a choice of setting the reduced d_{max} to either $\overline{d}^{max}$ or $\overline{d}^{avg}$ and the d_{min} remains at 2. Having the reduced d_{max} set to 2 satisfies us that we do not cut off any region of the sparser or lightweight solution space in which the optimal solution may lie. The choice for the value of the reduced d_{max} is user determined and varies with the network size. However, a reduced d_{max} set to $\overline{d}^{avg}$ forms a tighter upper bound and in some cases, seems to improve the speed of convergence of the heuristic to the solution. Having reduced our search window, we sweep back and forth (pare-up and pare-down) within this window until the stop criteria are satisfied.

The process of pare up and pare down provide the key to any cost improvement with this heuristic. Edges may be added and removed randomly or with deterministic criterion. Deterministic criteria we use for pare-down include:

a. Removing the edge with the least amount of total capacity placed,

b. Removing the edge with the least amount of spare capacity placed,

c. Removing the edge with the least amount of working capacity placed,

d. Removing the most expensive (longest) edge in the network

And for pare-up:

e. Adding the cheapest (shortest) candidate edge not yet in the network.

We intersperse the deterministic edge selection criteria with a random selection of edge for addition or removal to avoid being stuck in local minimum. Branch exchange routines are also included in between sweeps to provide a toggling or mutation of the solution. The implementation is such that after each sweep (pare-down and pare-up), there is a random number between a minimum of 1 and maximum of 3 edge exchanges before the next sweep. The nodal degree of the graph remains unchanged. The pseudocode for this is:

```
int numberOfExchanges = randomNumber (in the range [1, 3])
for (i = 0; i < numberOfSweeps; i++)
    pareDown;
    pareUp
    for (x = 0; x < numberOfExchanges; x++)
        exchangeEdges;
```

We have found it helpful to employ a taboo mechanism in the random additions or removals such that an edge removed cannot be added until a certain number of iterations (both edge additions and removals). Similarly, an edge added cannot be removed until a certain number of iterations. The number of iterations forms the taboo window size (or taboo period). This reduces the probability of having the same graph repeated. For example, if we have a window size of x, this means that an edge removed cannot be added to the graph until x edges have been removed. Similarly, an edge added to the graph cannot be removed until x edges have been added. The window size is set to be small for small problem sizes and could increase with the problem size.

The pseudocode is as follows:

```
int windowSize = x;
tabooList addedEdges, removedEdges;
pareDown:
select edge for removal from graph;
if edge is not taboo (included in addedEdges)
    remove edge from graph;
    add edge to removedEdges
    if (size of removedEdges == window Size)
        remove oldest edge from removedEdges.

pareUp:
select edge for addition to graph;
if edge is not taboo (included in removedEdges)
    add edge to graph
    add edge to addedEdges
    if (size of addedEdges == window Size)
        remove oldest edge from addedEdges.
```

The stopping criterion we used is a pre-set time limit. We set either an overall time for the whole process, or a specified duration for sweeping within the narrowed search window, or a specified number of sweeps. The default option is making three initial sweeps and then six sweeps over the narrowed search region. The user determines the number of sweeps for each phase. Figure 43 shows the pare-up/pare-down routines and the composite Sweep routine is shown in Figure 44. Figure 45 shows the flow diagram for the Sweep search heuristic.

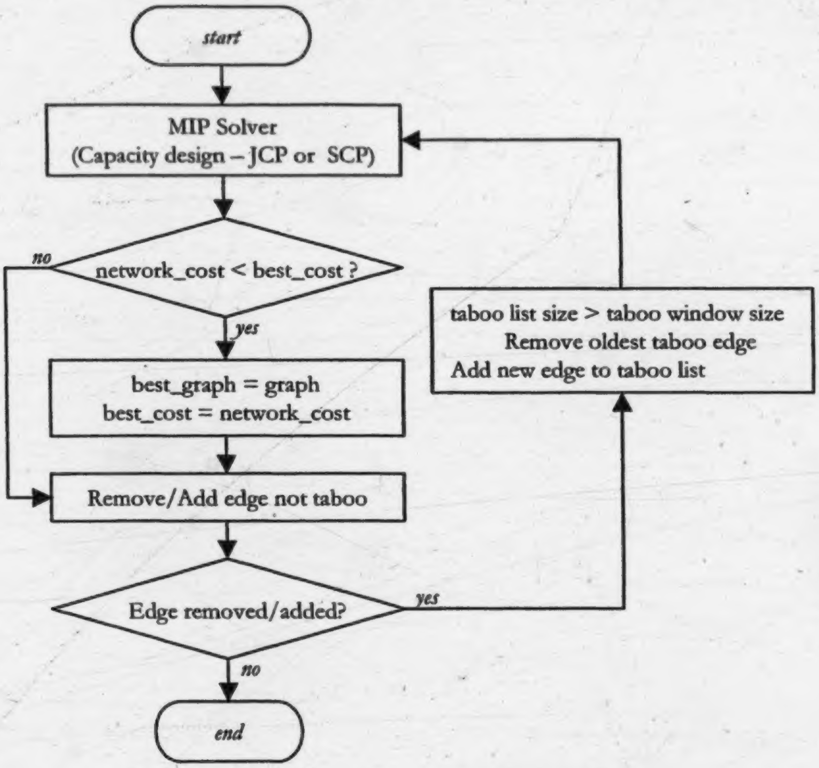

start
MIP Solver
(Capacity design – JCP or SCP)
network_cost < best_cost ?
no
yes
best_graph = graph
best_cost = network_cost
Remove/Add edge not taboo
Edge removed/added?
yes
no
end
taboo list size > taboo window size
Remove oldest taboo edge
Add new edge to taboo list

Figure 43 – The PARE-UP/PARE-DOWN routine

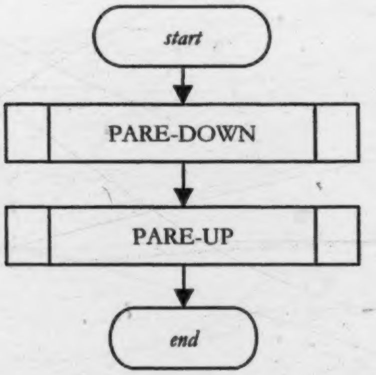

start
PARE-DOWN
PARE-UP
end

Figure 44 – The Sweep routine

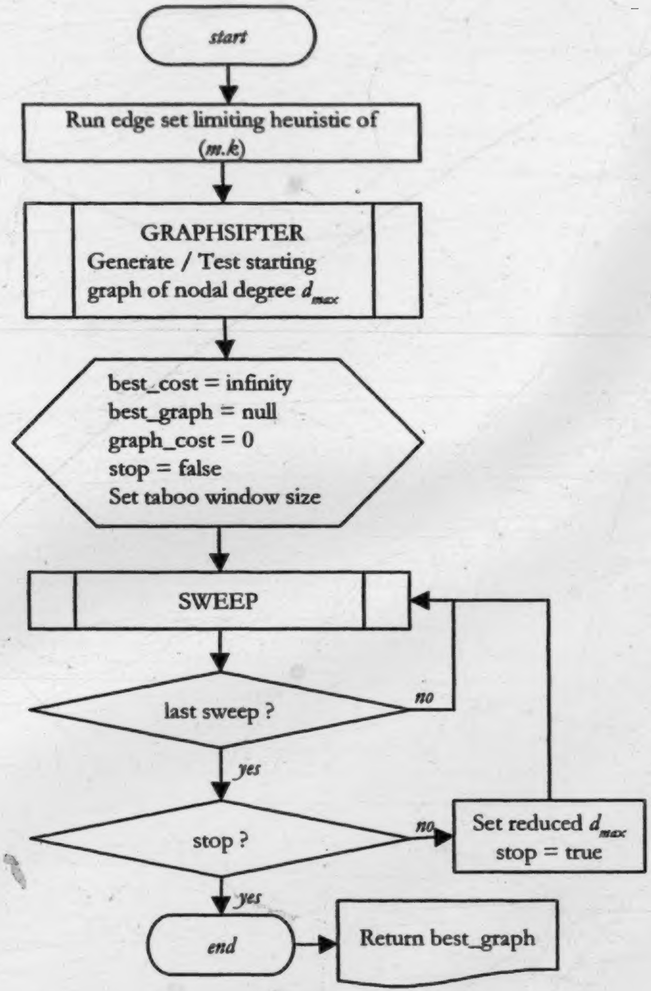

start
Run edge set limiting heuristic of (m.k)
GRAPHSIFTER
Generate / Test starting graph of nodal degree d_{max}
best_cost = infinity
best_graph = null
graph_cost = 0
stop = false
Set taboo window size
SWEEP
last sweep ?
no
yes
stop ?
no
Set reduced d_{max}
stop = true
yes
end
Return best_graph

Figure 45 – The Sweep Search flow diagram

Test results for the Sweep Search heuristic will be presented in Chapter 7.

6.4 ITERATIVE SAMPLING HEURISTIC

The Sweep Search heuristic proposed topologies for which routing/capacity designs were carried out. In this sense, the MTRS problem has been reduced to the more traditional network design problem of capacity placement given a fixed topology. Suppose the topology is not fixed. Then each edge vector generated is taken to be a specification of candidate edges, rather than the description of a particular graph topology. The MTRS MIP formulation is solved for this reduced set of edges, rather than the universal edge set. This is the idea exploited in the Iterative Sampling heuristic – to solve the MTRS problem on small candidate edge sets. For the Iterative Sampling heuristic, the steps are:

1. Obtain a starting graph G as the Minimum Spanning Tree based on the candidate edge set $E_{universe}$. This uses the MST graph generator described in Chapter 4. G is made bi-connected by applying the repair heuristic and the set of edges that describe G is noted as E_G.
2. Obtain $E_{iterate}$ as the set of edges from $E_{universe}$ not included in E_G ($E_{iterate} = E_{universe} - E_G$).
3. Solve the MTRS MIP problem with E_G as the candidate edge set. The set of edges in the optimal solution chosen from the candidate edge set E_G is noted as $E_{opt,G}$.
4. Add an edge $i \in E_{iterate}$ to E_G such that $E_{G'} = E_{opt,G} \cup i$ to describe an augmented edge set for graph G'. The MTRS MIP problem is then solved with $E_{G'}$ as the candidate edge set to give $E_{opt,G'}$. We do this until each edge $i \in E_{iterate}$ has been included as part of the candidate edge set.

The premise for this heuristic is as follows:

Suppose we have an edge set $E_G = \{e_1, e_2, e_3, \ldots.., e_m\}$. If the edge $e_1 \in E_G$ is not included in the optimal solution $E_{opt,G}$, we remove edge e_1 from any further inclusion in the edge set describing the overall optimal solution $\left(E_{opt}\right)$ for the network. In other words, we remove edge e_1 from $E_{universe}$. This follows from our interpretation that edge e_1 was removed because the edge set $\{E_G - e_1\}$ give a better solution. Therefore, as long as the edge set $\{E_G - e_1\}$ is included in the solution, we do not suffer any loss by ignoring edge e_1.

A resulting situation that may be possible is that given the edge set $E_G - e_1$, edge e_1 may not be included in $E_{opt,G}$ but if an edge e_x $(e_x \notin E_G)$ is now included as part of E_G, then edge e_1 may have been included in $E_{opt,G}$ and edge e_2 $(e_2 \in E_{opt,G})$ removed. Therefore, the edge set $E_{opt,G} = \{e_x, e_1, e_3, ..., e_m\}$ would be better than the edge set $E_{opt,G} = \{e_2, e_3, ..., e_m\}$. Unfortunately, edge e_1 had been removed from consideration before edge e_x was introduced. Therefore, the former edge set is not evaluated.

To reduce the probability of this occurring, we need to include as many potential edges as possible. However, we also need to limit the size of the candidate edge set included in the problem instance. We therefore set the start topology to have an average degree in the range $3 \le \bar{d} \le 4$ with smaller problem sizes having a higher nodal degree than larger problem sizes. Increasing the size of the candidate edge set comes with a trade-off in exponential growth in run-time with the increase in edges.

For example, we can solve a 10-node, 19-span problem by starting with a topology of 15 edges (average nodal degree of 3) and then, in 4 iterations we add a new edge each time and solve the MTRS problem for the reduced edge set. We achieve a run-time of 24.27 seconds. For the same problem, if we include all 19 edges in the problem at the same time (giving a graph with average nodal degree 3.8), we have a run-time of 169.47 seconds - about 700% increase in run-time. We find that though there are more iterations of solving the MTRS problem, the iterations run faster because the number of edges included in each problem is smaller. This is simply a divide-and-conquer technique.

Using the language of the local search optimization techniques, we define our start topology as a feasible solution x. The neighborhood of x, $N(x)$, is obtained by adding edges to x. In each iteration step the edge set included or admitted in the problem is different, thus for iteration $(n+1)$, the edge set under consideration is different from the edge set included in step (n) or step $(n+2)$. If we denote the optimal graph solution at step (n) as $G^*(n)$, the cost of the solution at step (n) as $c(n)$, then ideally, the strategy of the iterative sampling heuristic guarantees that $c(n) \geq c(n+1)$ (or $c(n+1) \leq c(n)$). This follows from the fact that the optimal edge set from step (n) is a complete subset of the edge set at step $(n+1)$ so the MTRS solution cannot be worse if solved to optimality within the candidate edge set. The solver would only accept $G^*(n+1)$ $\left(G^*(n+1) = G^*(n) + e\right.$ where e is the edge added to $G^*(n)$) only if $c(n+1) \leq c(n)$, else $G^*(n)$ is returned as the optimal solution from step $(n+1)$.

However, situations do arise in practice where we have $c(n+1) > c(n)$ if a significant MIP gap is set for the CPLEX solver. The MIP gap is the mixed integer optimality tolerance gap. If the MIP gap is set to $X\%$, what this means is that the solution we obtain is guaranteed to be within $X\%$ of the truly optimal solution (which we can get by setting the MIP gap to 0%). We can, therefore, say that strategy of the iterative sampling heuristic guarantees that $c(n+1) - c(n) \leq X\% \times c(n)$. There is a trade-off between the run-time and the MIP gap - a decrease in the MIP gap results in an increase in run-time and may not be worth it.

For the test networks we used, the start topology (based on a repaired minimum spanning tree graph of the network) contains a good portion of the edges that make up the optimal edge set. In fact, for COST239-6node to COST239-9node networks, the start topology contained all the edges of E_{opt}. The first iteration found the optimal solution and therefore, was not improved on for all other iterations. Figure 46 shows the convergence process to the optimal solution for two runs of the iterative sampling heuristic for the COST239-11n network (used with the published demand pattern for the network).

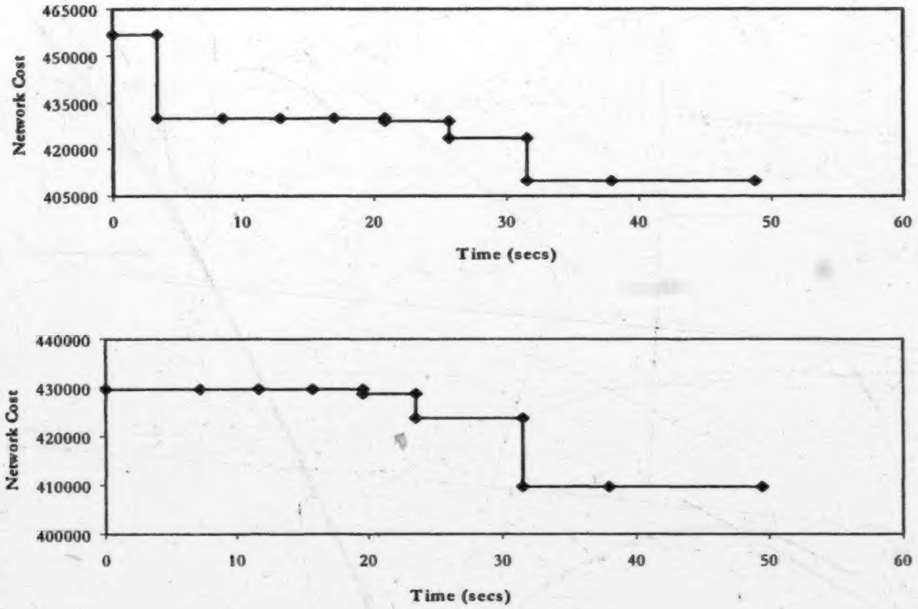
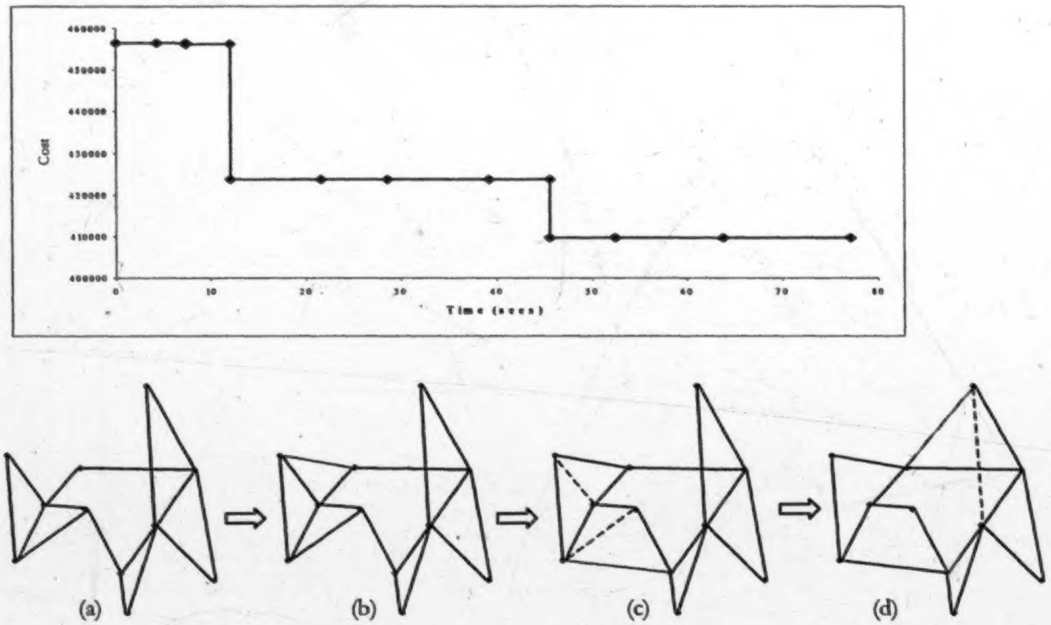


Figure 46 – Convergence on solution for iterative sampling

For the two runs shown in Figure 46, the starting topologies are different (indicated by the different network costs at time = 0 seconds). Each change in cost represents a change in optimal topology. Two different roads but heading to the same destination, in approximately the same time frame. The idea behind using the iterative sampling heuristic is one of constant improvement of the starting solution since each time an edge is added, the MTRS MIP solver cannot choose a solution with a cost higher than the cost of the starting solution.

Figure 47 shows the changes in topology with edge additions for the Iterative Sampling heuristic for the COST239-11n network. Applying the edge limiting heuristic, we reduce the problem size from 11-nodes, 55-spans to one of 11-nodes, 26-spans. We obtain a starting topology as a repaired MST graph having 17 edges (degree of 3.09) and 9 edges form $E_{universe}$ for this problem. The topology shown in Figure 47a is the optimal topology selected from the initial edge set of 17 edges with degree 2.91 (16 edges). Successively adding edges from $E_{universe}$, we have the run-time sequence shown below.



(a) (b) (c) (d)

Figure 47 – Topology changes with edge additions (1)

Topology changes occur only when an edge added in the sampling process is used in the solution. Such changes occur only if the new edge is included in the optimal solution along with the existing edges from a prior sampling solution (as shown in Figure 47b) or the new edge causes the removal of some existing edges (as shown in Figure 47c, d). The broken lines indicate edges which were part of a prior sampling solution but were removed from the current solution by the addition of a new edge.

Figure 48 shows another run of the Iterative Sampling heuristic for the same problem with the associated topology changes.

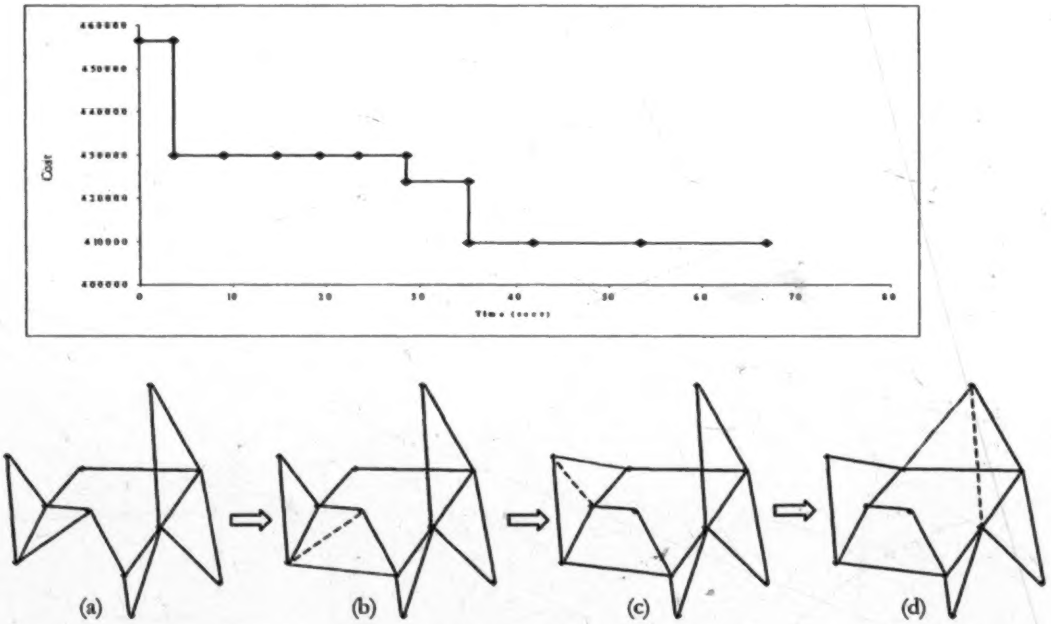


Figure 48 – Topology changes with edge additions (2)

Figure 49 shows the flow diagram for the Iterative Sampling heuristic.

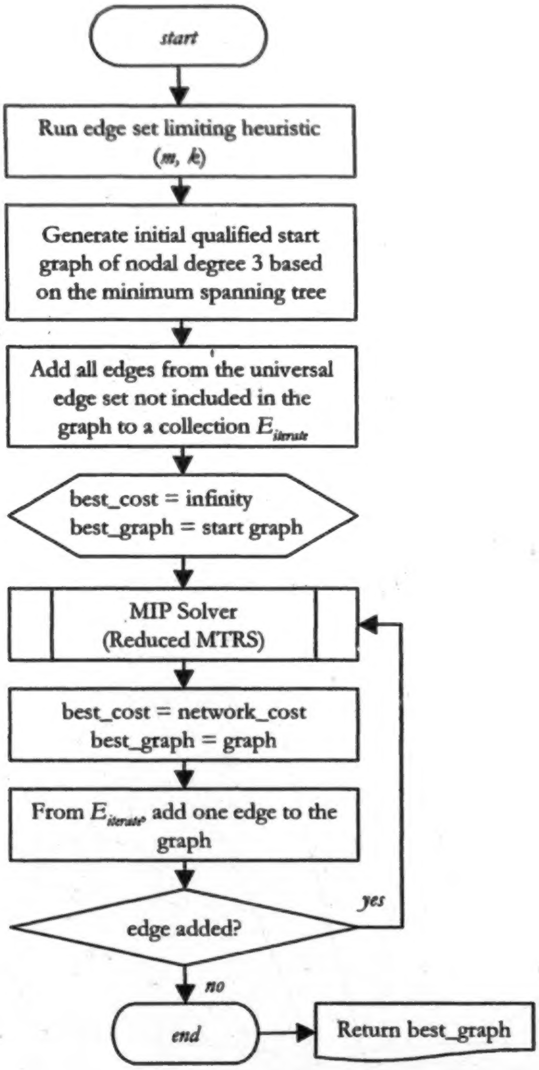

start
Run edge set limiting heuristic (m, k)
Generate initial qualified start graph of nodal degree 3 based on the minimum spanning tree
Add all edges from the universal edge set not included in the graph to a collection $E_{iterate}$
best_cost = infinity
best_graph = start graph
MIP Solver (Reduced MTRS)
best_cost = network_cost
best_graph = graph
From $E_{iterate}$, add one edge to the graph
edge added?
yes
no
end
Return best_graph

Figure 49 – The Iterative Sampling flow diagram

6.5 COMPLEXITY ANALYSIS OF THE HEURISTICS

First for the edge limiting heuristic, the complexity analysis is as follows:

First, the edges are sorted. The sort routine is a native Java sort routine, which is a modified mergesort algorithm and gives a guaranteed performance of $s\log(s)$ where s is the number of edges in the problem (typically $|N(N-1)/2|$). On nearly sorted lists, the performance of the sort algorithm can approach linear performance. Next, m edges are selected (process is $\Theta(1)$) leaving $s-m$ edges for further processing. For each of the $s-m$ edges, we check if the nodal degree of its end nodes is greater than some value *k*. Again, this process is of $\Theta(1)$. Thus, the complexity of the edge limiting heuristic is $O(s\log(s)+s-m) \cong O(s\log(s))$.

From [Kers93], we have the complexity of a typical routing algorithm that considers all node pairs given as $O(N^3)$. Each of the heuristics involves repetitions of the MIP capacity design thus they are at least $O(N^3)$.

For the Sweep Search heuristic, the complexity analysis is as follows:

Starting with a graph G having m edges and n nodes, the worst-case complexity of edge additions is $O(mc)$ if a taboo mechanism of window size c is employed and $\Theta(1)$ otherwise. This is because when the taboo list is used, a worst-case scenario would be when each edge selected for inclusion in the current graph has to be evaluated against every element in the taboo list. If the taboo list is not used, then each edge selected is simply added to the edge set of the graph.

Edge removals have a worst-case complexity of $\Theta(m)$. This describes the case when no edge in the graph can be removed (all edges are tried, else there would be some edge that could be removed and was not tried). Edge removals have a best-case complexity of $\Theta(1)$.

The pare-down and pare-up routines would have an average of $m-(n+2)$ repetitions of $O(N^3)$, giving a total complexity of about $O\left(c\left(m-(n+2)\times N^3\right)\right)$ where c is the total number of times both routines are called. This is $O(N^5)$

For the Iterative Sampling heuristic, the complexity analysis is as follows:

We have reduced the size of the edge set to some new value $s \geq 0.2N(N-1)$. If m edges are initially admitted to the start topology, then $s-m$ edges are excluded and therefore, we have $s-m$ iterations of capacity design routine giving a complexity of $O\left((s-m)\times N^3\right)$. This is typically $O\left(N^4\right)$.

7. TESTING AND RESULTS

7.1 INTRODUCTION

We conduct two series of final tests classed by the network size. The first series of test problems were based on seven, eight and nine-node problem instances. The spatial layout of nodes in the network test cases were obtained by placing the nodes at random (x, y) coordinates on a plane and the length of each edge in the network is the Euclidean distance between its end node pair (networks 9n36s1 and 9n36s2 networks were used in the study in [GrDo01] which allows for comparative analysis of results). For the test cases, we set the values of the edge-to-unit capacity cost ratio Ω to 25, $F_{ij} = \Omega l_{ij}$ and $c_{ij} = l_{ij}$ (where for span (i, j), F_{ij} is the establishment cost, l_{ij} is the length, and c_{ij} is the cost of adding a unit of capacity).

The demand matrix for each network has a non-zero demand value for every O-D pair. For the seven and eight-node networks, we generate two demand matrices for each network: an inverse distance weighted gravity model and a uniform random distribution demand pattern. For the nine-node network, we use only the gravity model. To generate the random pattern, each O-D pair is assigned a demand intensity value from a uniform random distribution of integers in the range [1, 15] regardless of distance or nodal degree. For the gravity model, we used the relationship $k.\deg(O).\deg(D)/L(O,D)$ where $2 \geq k \leq 50$ is chosen to simulate high network traffic compared to the random model while keeping the maximum demand quantity between any O-D pair at 20 (networks in which nodes have a greater spatial separation have a lower value of *k*).

We use both demand pattern models to enable us to test the robustness of the heuristics, especially the edge limiting heuristic under varying demand loads. We want to be able to answer the question: Are good potential candidates (edges that could be admitted into a solution) being eliminated by the edge limiting heuristic?

These notionally small problems turn out to be quite computationally intensive, especially the eight and nine-node problems because of the completeness of the candidate edge sets and the demand matrices.

The second series of tests were based on ten, eleven, fifteen, twenty, twenty-three and twenty-six node problem instances. All networks are random networks obtained in the manner earlier described. The 10n45s2, 11n55s1, 20n88s, 23n104s and 26n127s networks were also used in the study in [GrDo01]. Networks 10n45s2 and 11n55s1 are used as presented in the study while networks 20n88s, 23n104s and 26n127s have an additional 8, 12 and 23 spans respectively. Network C239-11n is the previously described COST239-11n network. We use the inverse distance weighted gravity model of demand pattern to generate a demand matrices for each network where every O-D pair exchanges a non-zero demand quantity. So while the edge set for the networks are not complete, they do involve a complete demand pattern. This makes the problem computationally intensive. The edge-to-unit capacity cost ratio Ω is also set to 25, $F_{ij} = \Omega l_{ij}$ and $c_{ij} = l_{ij}$.

For both the large and small test networks, we solve the MTRS optimization problem time-limited to four hours for each network in the second series of tests for the networks with the edges set as given in each problem. Then we apply the edge limiting heuristic and re-solve a time limited MTRS optimization for each problem (reduced MTRS). The edge limiting heuristic we apply is to set $m = 0.3 \times |N|.d_{\max}$ and $k = \{3,4\}$ unless stated otherwise. We compare the solutions (cost and time) for both. The number of edge candidates for each test network before and after the edge limiting heuristic is summarized in Table 15. If we look at the column, we observe that for some networks (for example, the 8-node and 10-

node networks highlighted), given the complete edge set, the edge limiting heuristic did not leave the same number of edges. This shows a responsiveness or adaptability of the algorithm to different problems.

Table 15 – Size of candidate edge sets used for testing ($m = 0.3\times|N|.d_{\max}$, $k = 3$)

Test Network	$E_{complete}$	$E_{reduced}$	Test Network	$E_{complete}$	$E_{reduced}$
7n21s1	21	13	10n45s2	45	19
7n21s2	21	13	11n55s1	55	26
8n28s1	28	14	C239-11n	55	26
8n28s2	28	15	15n59s	59	34
9n36s1	36	17	20n88s	88	45
9n36s2	36	17	23n104s	104	52
10n45s1	45	18	26n127s	127	59

We compare the performance of the Sweep Search and Iterative Sampling heuristics (in time and design cost) to the reduced MTRS solutions because the heuristics admit and operate on the same edge set (the first step in each heuristics is edge set limitation).

In the absence of a strictly optimal reference solution for the large test networks, we use the 3-step heuristic mentioned earlier to further evaluate the performance of the heuristics on the problems.

7.2 EXPERIMENTAL RESULTS

Test Series 1:

This test series involve small test networks with node size less than ten. We started with these problem sizes because we expected to obtain optimal reference solutions for them with which we can evaluate the performance of the heuristics.

Table 16 summarizes the results for the series 1 tests. The test network name (for example, 7n21s1-G) indicates the number of nodes (in this case, 7), the cardinality of the candidate edge set admitted into the problem (e.g. there are 21 candidate edges for a 7-node network), a number to distinguish between different 7-node network problems (e.g. 1) and the demand model used, where G represents the gravity model and R represents the random demand pattern.

Table 16 – Complete and reduced edge set MTRS solutions for test series 1 problems

Test Network	MTRS - Complete Edge Sets						MTRS - Reduced Edge Sets						Δ (%)
	$\overline{d}$	Total Cap.	Total Cost	Time (secs)	Edge /Cap	Notes	$\overline{d}$	Total Cap.	Total Cost	Time (secs)	Edge /Cap	Notes	
7n21s1-G	2.86	310	147069.7	1116.91	0.81	FT	2.86	310	147069.7	7.95	0.81	FT	0.0%
7n21s1-R	2.86	324	152301.4	1973.53	0.76	FT	2.86	324	152301.4	7.99	0.76	FT	0.0%
7n21s2-G	3.14	341	132249.6	910.07	0.82	FT	3.14	341	132249.6	9.84	0.82	FT	0.0%
7n21s2-R	3.14	307	128077.6	1363.79	0.88	FT	3.14	307	128077.6	7.77	0.88	FT	0.0%
8n28s1-G	3.00	611	191315.8	14821.3	0.51	FT	3.00	611	191315.8	4.78	0.51	FT	0.0%
8n28s1-R	3.00	438	158822.0	30949.8	0.69	FT	3.00	438	158822.0	7.72	0.69	FT	0.0%
8n28s2-G	3.00	390	126475.9	12704.6	0.82	FT	3.00	390	126475.9	21.43	0.82	FT	0.0%
8n28s2-R	3.25	473	149738.2	14165.4	0.69	FT	3.25	473	149738.2	17.33	0.69	FT	0.0%
9n36s1-G	4.00	537	32691.0	28282.4	0.85	TL (4 h)	3.56	599	32257.0	85.20	0.68	FT	-1.33%
9n36s2-G	3.56	554	34161.0	29912.8	0.74	TL (4 h)	3.11	599	32654.0	59.83	0.58	FT	-4.41%

Table 16 shows some details of the solutions obtained from solving the MTRS problem first with the universe of edge candidates and then, with the candidate edge set obtained from applying the edge limitation heuristic. For both reported MTRS problems, the first column gives the degree of the solution graph obtained for the network. Column 2 shows the amount of capacity placed in the network. The third column shows the time taken to solve the problem. We had earlier stated that the run time we report when we use a multi-processor system is the solution time – the sum total of the amount of time each processor spent on the problem – which would be comparable to the amount of time a uni-processor platform would take for the same problem. When we set a time limit however, it is only a limit of how much real time is spent on the problem, and is less than the actual total CPU solution time. The last two rows of Table 16 report the test data for the nine-node networks. We see that there was a time limit of 4 hours, yet the actual solution time for each problem is over 7 hours.

The *Edge/Cap* column shows the ratio of the total edge costs to the total capacity costs for the solution. A value of edge/cap less than 1 means that the total capacity costs contribute more than the fixed costs to the total network costs while a value greater than 1 would mean that the fixed costs dominate the network costs. A value approximately or in the region of 1 suggests that both costs contribute almost equally to the network cost (i.e. the notion of $\Omega \approx \Omega_c$ is apparent. In the notes column, we identify how CPLEX terminated and therefore the type of solution obtained. "FT" results were solved to optimality (full termination). "TL" results are the best feasible results obtained within the time limit but the problem was not solved to optimality. In the Δ column, we report the difference between the network cost obtained from solving the MTRS problem with the reduced edge set versus the attempt at solving the problem with the complete edge set. A negative Δ value says that the solution cost for the reduced edge set is lower than that obtained for the complete edge set and a positive Δ means that the reduced edge set has a higher cost. We see a negative Δ value only when we have a time-limited (non-optimal) run of the MTRS problem. A gap of 0% means that both solutions are same and the enhanced performance lies in the run-time.

Let us now walk through the test details for networks 8n28s2-R and 9n36s2-G as examples to explain the aspects of the table and begin a discussion of the results. Network

8n28s2-R is an 8-node network having the complete set of 28 candidate edges. The demand pattern used is a random pattern where for each O-D pair, a random integer value in the range [1, 15] is assigned as the demand quantity exchanged.

First we check the Δ to determine the quality of the solution. We see that the Δ is 0% telling us that the solutions to the MTRS problem solved on the complete candidate edge set and the problem solved on the reduced candidate edge set are the same. Then we check the performance in terms of run-time. Given the complete edge set of size 28, the MTRS solver takes 14165.8 seconds (3.93 hours) to solve the problem to optimality. In contrast, given the reduced edge set of size 15, the solver finds the optimal solution in 17.33 seconds. The edge/cap value is 0.69 suggesting that the total costs are dominated slightly by the capacity costs (since the capacity costs are higher), but both edges and capacity are important in the solution.

The second network 9n36s2-G is a 9-node network with all 36 candidate edges and the demand pattern used was the gravity model. After a four-hour time-limited run of the complete problem, the best cost network solution the MTRS solver found was 4.41% higher than the solution found with the reduced candidate edge set in 59.8 seconds. It is very probable that the solution to the MTRS-Reduced edge set problem is the optimal solution (given the results of FT solutions). The solution after four hours given the complete edge set contains more edges that the reduced edge set solution, thus verifying our continuing belief that the MIP solver spends so much time searching in high weight (edge count) graphs for a solution, making an investment of time for capacity considerations for these graphs that could be considered a waste. The edge/cap value of this network also shows the dominance of the capacity cost on the total network costs. This is same for all the networks considered and is attributed to the intensity of the demand matrices we use and the value of Ω.

We see from the results of the MIP solver using the reduced candidate edge set that we have been able to guide the solver into a region of combinatorial search containing only viable edge candidates, of which the optimal solution edge set is usually a subset. The conclusion is that we have not compromised the solution quality by the limiting heuristic, but achieved a performance enhancement by decreasing the run-time.

Let us now consider how the Iterative sampling and Sweep Search heuristics compare to these results. We compare each heuristic directly with the MTRS – Reduced Edge set solutions (the latter test results are re-stated from Table 16).

Table 17 – Reduced MTRS vs. Iterative Sampling results

	MTRS - Reduced Edge Set					**MTRS - Iterative Sampling**					
Test Network	$\bar{d}$	**Total Cap.**	**Total Cost**	**Time (secs)**	**Notes**	$\bar{d}$	**Total Cap.**	**Total Cost**	**Time (secs)**	**Notes**	**Δ (%)**
7n21s1-G	2.86	310	147069.7	7.95	FT	2.86	310	147069.7	5.92	FT	0.00%
7n21s1-R	2.86	324	152301.4	7.99	FT	2.86	324	152301.4	6.04	FT	0.00%
7n21s2-G	3.14	341	132249.6	9.84	FT	3.14	341	132249.6	6.71	FT	0.00%
7n21s2-R	3.14	307	128077.6	7.77	FT	3.14	307	128077.6	5.14	FT	0.00%
8n28s1-G	3.00	611	191315.8	4.78	FT	3.00	611	191315.8	4.48	FT	0.00%
8n28s1-R	3.00	438	158822	7.72	FT	3.00	438	158822	6.71	FT	0.00%
8n28s2-G	3.00	390	126475.9	21.43	FT	3.00	390	126475.9	14.23	FT	0.00%
8n28s2-R	3.25	473	149738.2	17.33	FT	3.25	473	149738.2	11.23	FT	0.00%
9n36s1-G	3.56	599	32257	85.2	FT	3.56	599	32257	39.7	FT	0.00%
9n36s2-G	3.11	599	32654	59.83	FT	3.11	599	32654	22.12	FT	0.00%

Δ as reported here is the difference in % between solution costs for the iterative sampling heuristic with the solutions costs for the reduced MTRS problem. First, we check the quality of solutions obtained by the iterative sampling heuristic as determined by Δ. A glance shows that it is zero for all problems, showing that the optimal solution was found each time. Next, we compare the run time. We see that the performance of the iterative sampling heuristic in these small network sizes is quite comparable, and seems to be better than that of the reduced MTRS because of its slightly lower run times. In the following test series 2, we will see its performance in large network sizes.

Our implementation of Sweep Search heuristic allows only for a user-defined stopping criterion – search time limit or a pre-defined number of sweeps. We specify two initial sweeps because the network sizes are small (2 pare-up and pare-down routines each) and the reduced or narrowed d_{max} value is set to $\overline{d}^{avg}$. Within the narrowed search region, we specify 2 sweeps (larger networks may require more). We shall limit ourselves to comparing the quality of the solution obtained after a run along with the run-time. We illustrate the sweep search heuristic results with an example from the 7n21s2-G network.

The pare-up and pare-down process uses any combination of the criteria earlier mentioned. For the pare-down routine, the choice of edge to be removed is made by a random choice of any of the four deterministic edge selection criteria previously mentioned or a randomly chosen edge. For the pare-up routine, the choice of edge to be added is a random choice between the deterministic edge selection criterion (the shortest edge from the universal candidate edge set not yet included in the edge set for the graph), a random edge which is not taboo, or a random choice ignoring the taboo restriction.

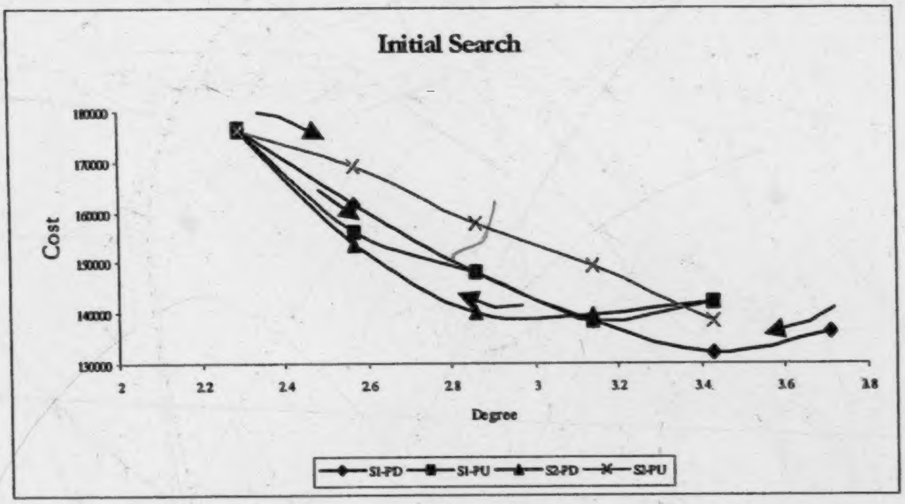


Figure 50 – Initial search of 2 sweeps from d_{max} to d_{min} for 7n21s2-G

Figure 50 shows the initial sweep search through the region of the maximum and minimum nodal degree the network could have. The edge limiting heuristic reduced the candidate edge set from 21 edges to 13 edges. Therefore, the default value of d_{max} for this network is 3.71. The arrows show the direction of sweep. The initial sweep period took approximately 24 seconds and the best-cost graph obtained within this period has a cost of 132276.2 and nodal degree 3.43. (Table 16 shows that the optimal solution for this test case has $\overline{d}_{opt} = 3.14$ and the network cost is 132249.6).

The reduced d_{max} is set to $\overline{d}^{avg}$ which 3.43. Figure 51 shows a search through the narrowed search region. The initial best-cost graph was not improved on within the narrowed search window and this search took approximately 23 seconds.

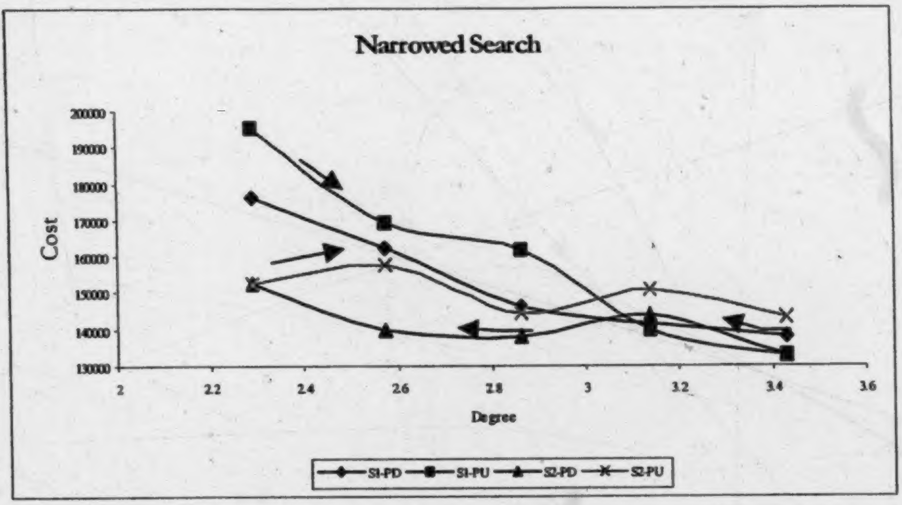


Figure 51 – Narrowed search of 4 sweeps from d_{max} to d_{min} for 7n21s2-G

In this run, we were not able to arrive at the optimal solution for this test problem. However in some runs of the same problem instance, we are able to arrive at the optimal solution.

Setting the d_{min} and d_{max} for the narrowed search region as 2 and $\{\overline{d}^{max}, \overline{d}^{avg}\}$ respectively gives a wide search region as shown in Figure 51. we try to narrow this region by increasing the value of the reduced d_{min}. This leads to alternate criteria for setting the reduced d_{max}.

During the initial sweep, the cost of each network found is recorded against its average nodal degree. At the end, the average cost for each nodal degree is found and this value is plotted against the degree. Suppose the best network cost found after the initial sweep is C. We then set an upper cost limit for the search region as $C\alpha$ where $\alpha \geq 1$. Where the function $y = C\alpha$ intersects the plot of average cost vs. degree, we have the value of the reduced nodal degree limits. This is illustrated in Figure 52.

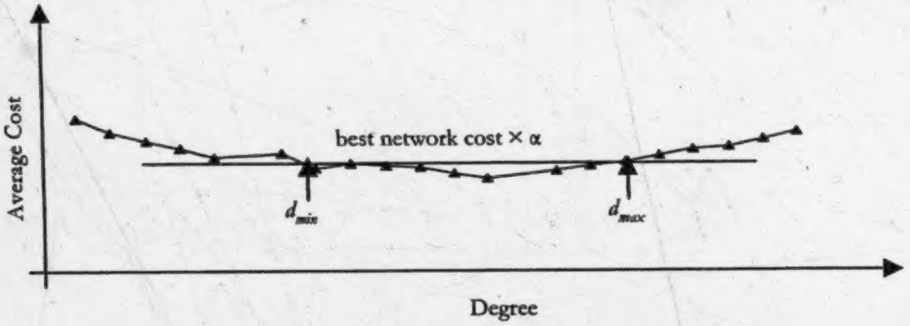


Figure 52 – Ideal performance of criteria for narrowing search region

Using this modification for the Sweep Search heuristic, we see the following results.

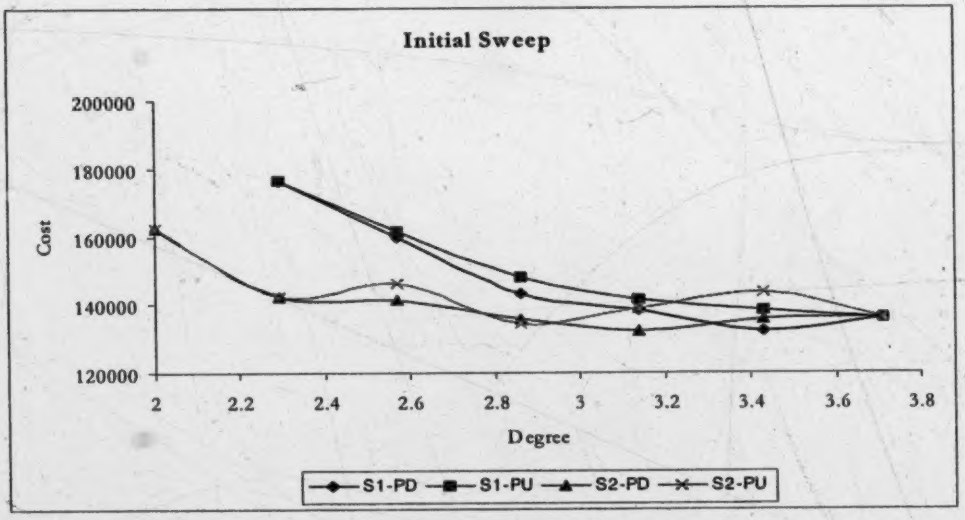


Figure 53 - Initial search of 2 sweeps from d_{max} to d_{min} for 7n21s2-G

The average cost at each nodal degree between 2 and 3.71 is found and plotted against the degree as shown in Figure 54. α is set to 1.1. The reduced d_{max} remains at 3.71 because the function $C\alpha$ did not intersect with the cost curve at any upper nodal degree limit. The reduced d_{min} is set to 2.89 as indicated. Figure 55 shows the search within this narrowed search region and we were able to obtain the optimal solution in this run.

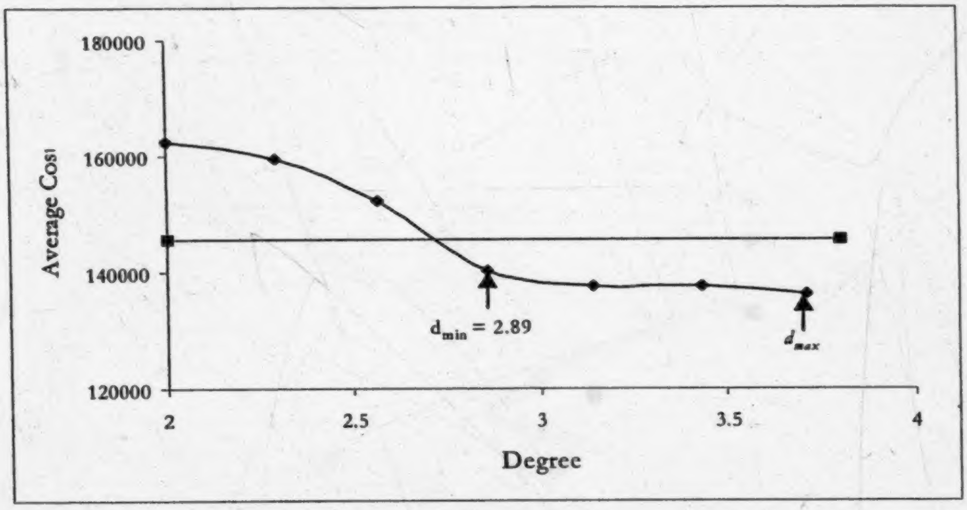


Figure 54 – Average cost versus nodal degree

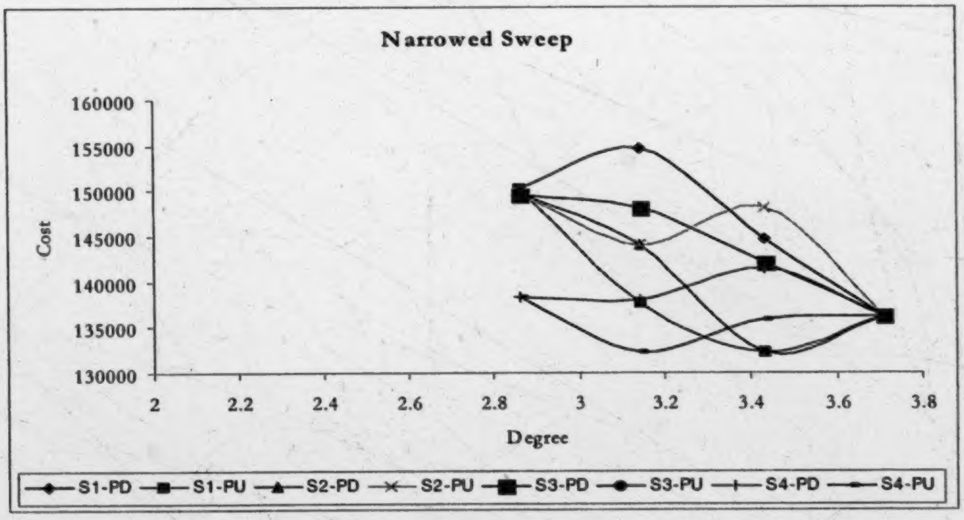


Figure 55 - Narrowed search of 4 sweeps from d_{max} to d_{min} for 7n21s2-G

Using some other networks, we compare the performance of the Iterative heuristic to the prior 3-step MIP heuristic [GrDo01]. Table 18 shows the details. Solutions from the reduced and complete MTRS problems are included as reference. "IF" results show that within a time limit, the solver was not able to find any feasible solution.

Table 18 – Iterative Sampling heuristic test results

Network	Full MTRS		MTRS (Red.)		Iterative Samp.		3-step (Red.)		3-step (Full)	
	Avg. time (secs)	Cost	Avg. time	Cost	Avg. time (secs)	Cost	Avg. time (secs)	Cost	Avg. time (secs)	Cost
COST239-7n	51.57	191358	2.83	191358	3.70	191358	2.97	191358	3.96	191358
COST239-8n	5721.06	224389	6.04	224389	6.71	224389	4.21	224389	6.55	224389
COST239-9n	333451.09	299673	17.81	299673	9.80	299673	5.9	304383	19.65	304383
9n36s4	409233.53	23912	45.17	23912	17.64	23912	12.35	24971	109.55	24971
COST239-10n	IF		190.3	372723	17.81	372723	13.34	378567	156.79	378567
COST239-11n	IF		3614.35	409702	63.04	409702	33.31	418057	543.34	418057
15n56s	IF		TL (4 h)	37770	990.33	37715	2193.22	38848	IF	

In Table 18, the highlighted portions show the best-cost test results for each test case using various solution methods. Where the network cost is the same for all methods, the best is determined by the method having the lowest run-time. For the 7- and 8-node networks, all solution methods find the best-cost network. For the 7-node network, solving the MTRS with the reduced candidate edge set provides a better performance in terms of shortest run-time. The 3-step heuristic run also on the reduced edge set provides a comparable performance and for the 8-node network, it gives the best result in terms of run-time. For the 9-, 10-, and 11-node networks, the iterative sampling heuristic gives the best performance in terms of time and total solution cost. The 3-step heuristic executes faster than the iterative sampling but for these test networks, fails to find a solution as good as the one obtained using the iterative sampling approach. For the 10- and 11-node problem given the full universe of edge candidates, the MTRS MIP formulation is not solved to termination within 10 days.

The 15n56s network is used as described in [GrDo01]. For this test case, the iterative sampling heuristic gives the best performance in terms of time and solution cost having a run-time of 16.5 minutes with Δ of -0.15% as compared to the cost of the MTRS problem solution (given a reduced candidate edge set) obtained within a 4-hour time limit of the problem (actual solution time is 57245.05 seconds).

We can conclude from our results that for networks with node sizes $|N| \leq 9$, the edge limiting heuristic followed by the MTRS MIP optimization suffices to produce optimal solutions within a reasonable amount of time. The use of a heuristic approach for problems of these sizes may not offer much performance improvement. We turn our attention to networks with $|N| \geq 10$ in the second test series 2.

Test Series 2:

In Test Series 1, we used test cases of size $|N| \leq 9$ to evaluate the heuristic presented in this work. In Test series 2, we will use only test cases of size $|N| \geq 10$. Our experience with problems of these sizes show us that solving the full problem to optimality is not expected to be possible within a practical time frame. We therefore try to obtain reference solutions by solving a time limited run of the MTRS complete and MTRS reduced edge set problems. We will then compare the performance of the heuristics against the best solutions obtained from these two reference solutions. Table 19 shows the details of the test results for the MTRS MIP formulations.

Table 19 – MTRS reference results for test series 2

Test Network	MTRS - Complete Edge Set						MTRS - Reduced Edge Set						Δ (%)
	$\bar{d}$	Total Cap.	Total Cost	Time (secs)	Edge /Cap	Note	$\bar{d}$	Total Cap.	Total Cost	Time (secs)	Edge/ Cap	Note	
10n45s1-G	3.2	433	164499.2	31659	0.81	TL (4 h)	2.80	451	156022.4	134	0.78	FT	-5.2%
10n45s2-G	3.4	523	27058.0	28423	0.76	TL (4 h)	3.00	566	25542.0	238	0.66	FT	-5.6%
11n55s1-G	4.18	604	36812.0	37877	0.82	TL (4 h)	3.64	657	34373.0	30539	0.77	TL (4 h)	-6.6%
C239-11n-G	3.27	556	453094.0	29753	0.88	TL (4 h)	3.09	589	439376.0	32011	0.74	TL (4 h)	-3.0%
15n59s-G	4.13	845	36046.0	29100	0.51	TL (4 h)	3.20	1088	30743.0	57285	0.57	TL (4 h)	-14.7%
20n88s-G	2.4	3754	242260.0	27493	0.69	TL (4 h)	3.70	1858	156356.0	56825	0.53	TL (4 h)	-35.5%
23n104s-G	IF					TL (4 h)	3.74	3354	218635.6	55578	0.33	TL (4 h)	
26n127s-G	IF					TL (4 h)	4.08	4114	243994.0	54978	0.33	TL (4 h)	

The format for the table is the same as we have seen earlier. The first column gives the network details – number of nodes, size of the universe of edge candidates and the type of demand pattern used (e.g. 23n104s-G). "TL", "FT" and "IF" are as explained before. The highlighted portion of Table 19 indicates the best-cost reference solution that we obtained.

From the results in Table 19, we again see the MTRS MIP solver bogged down in search space containing many edges because solutions of this nature are clearly more abundant. In general, we can see that limiting the edge set is the first step to attaining solutions that can be considered optimal or good approximations. Take for example, the row for network 20n88s-G in Table 19. We are able to obtain a value of Δ = -35.5% when we applying the edge limiting heuristic though the problem was not solved to optimality in both cases (both problems were solved with the same time limit). For network 15n59s, the solution after 4 hours with the un-reduced edge set has a degree of 4.13 (31 edges) whereas, a better solution (Δ = -14.7%) exists (found when the edge set is reduced) with a degree of 3.2 (24 edges).

We now evaluate the performance of the iterative sampling and sweep search heuristics.

7.2.1. ITERATIVE SAMPLING HEURISTIC RESULTS

Table 20 – Iterative Sampling heuristic results

Test Network	MTRS - Reduced Edge Set						MTRS - Iterative Sampling						Δ (%)
	$\bar{d}$	Total Cap.	Total Cost	Time (secs)	Edge/ Cap	Note	$\bar{d}$	Total Cap.	Total Cost	Time (secs)	Edge/ Cap	Note	
10n45s1-G	2.80	451	156022.4	134.6	0.78	FT	2.80	451	156022.4	21.42	0.78	FT	0.0%
10n45s2-G	3.00	566	25542.0	238.3	0.66	FT	3.00	566	25542.0	24.27	0.76	FT	0.0%
11n55s1-G	3.64	657	34373.0	30539	0.77	TL (4 h)	3.09	657	34378.0	209.42	0.56	FT	0.0%
C239-11n-G	3.09	589	439376.0	32011	0.74	TL (4 h)	2.91	613	445425.0	67.86	0.68	FT	1.4%
15n59s-G	3.20	1088	30743.0	57285	0.57	TL (4 h)	3.07	1156	30407.0	3273.93	0.52	FT	-1.1%
20n88s-G	3.70	1858	156356.0	56825	0.53	TL (4 h)	3.30	1989	149407.0	55715.98	0.42	FT	-4.4%
23n104s-G	3.74	3354	218635.6	55578	0.33	TL (4 h)	3.30	3871	223284.3	90079.57	0.25	AB	2.1%
26n127s-G	4.08	4114	243994.0	54977	0.33	TL (4 h)	-						

Table 20 shows the results of the iterative sampling heuristic results compared to the reduced edge set MTRS solutions. The iterative sampling heuristic starts off well, finding what we can consider to be the optimal solution for networks 10n45s1, 10n45s2 and 11n55s1. For these three problems, we consider the run-time fast and reasonable compared to the MTRS-Reduced edge set run-times for the same networks. For network 15n59s we had looked at earlier, the heuristic finds an even better solution with a degree of 3.07 (23 edges) that has Δ = -1.1% better than the best solution found before by the MTRS MIP solver with the reduced edge set within a time limit. The iterative sampling solution was found in ~55 minutes on the uni-processor computer platform while the MTRS MIP solution on the reduced edge set was obtained in an equivalent time of 15.9 hours.

For network C239-11n-G, the solution cost from the iterative sampling heuristic is 1.4% worse than the cost for the MTRS-Reduced edge problem. We consider this to be as a result of the earlier mentioned problem that the iterative heuristic could have – removing an edge at step (*n*) of the iteration process, if at step (*n+1*), the edge would have been viable.

The performance of the iterative sampling heuristic (in terms of run-time) quickly degrades however, as the network size gets larger resulting in the heuristic being aborted for the 23n104s network after a real solution time of 25 hours by the user ("AB" means aborted). At this time, the best found solution has $\Delta = 2.1\%$ worse than the reduced edge set MIP solution found within a four-hour time-limited run (the equivalent CPU processing time however is 15.4 hours).

Solving a time limited run of each iteration step is not meaningful to the success of this heuristic because at each step of the iteration, the edge set chosen has to be the optimal edge set from the edges under consideration. That is the only condition under which we feel can safely ignore step $(n+1)$, an edge that was not chosen (as part of the optimal edge set) in step (n). We assume that the run-time for this heuristic has an exponential growth rate with the problem size as shown in Figure 56. This hinders the effectiveness of applying this heuristic in problems of large network sizes. All run-times (except for the 23n network) are the actual times taken for a full-termination of the problem. Solution times for the 7- to 15-node problems are taken from Table 18 while solution times for the 20- and 23-node problem are taken from Table 20.

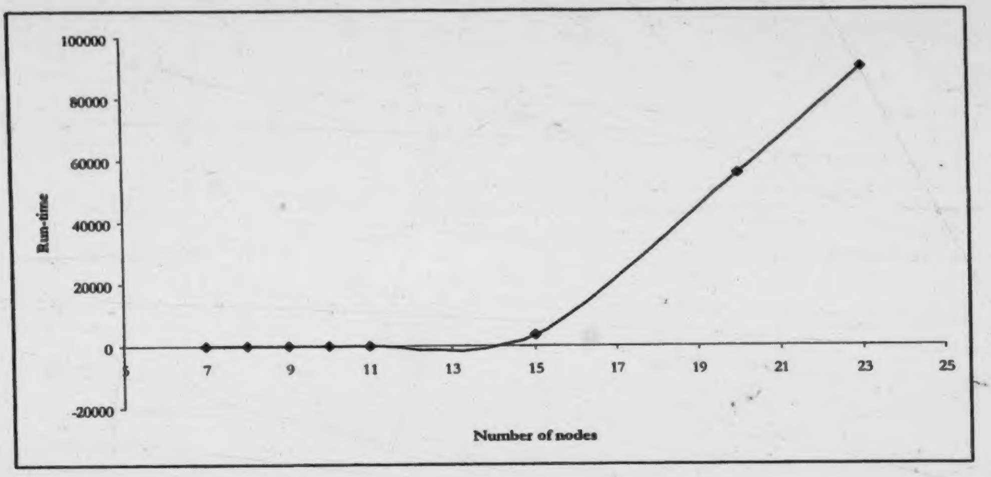


Figure 56 - Rate of growth of run time of Iterative sampling with problem size

7.2.2 SWEEP SEARCH RESULTS

We use the 15n59s network to assess the performance of the Sweep Search heuristic. Network 15n59s has maximum nodal degree of 7.87 because the edge set under consideration has a size of 59. When we apply the edge limiting heuristic, the edge set size is reduced to 34 leaving the network with a maximum nodal degree of 4.53. The d_{max} is then 4.53 and d_{min} is 2.

The initial number of sweeps was set to 5 and the sweep period lasted for 6782.2 seconds yielding the best-cost graph with a cost of 32053.0 with nodal degree of 3.6. The reduced d_{max} was then set to 3.73 and the number of sweeps through the narrowed region was set to 10. This second sweep period lasted for 6352.9 seconds (giving a total run-time of 13135.1 seconds) and improved on the best-cost graph solution found. The new solution was a topology of average nodal degree 3.07 and total network costs of 31147. Table 21 shows the best-found values during each sweep routine.

Table 21 – Progress of the Sweep Search heuristic for the 15n59s network

Initial Sweep			
Sweep	**Pare**	**Best Cost**	**Degree**
1	Down	32053	3.6
	Up	32106	3.47
2	Down	32701	3.33
	Up	32790	3.2
3	Down	33215	3.73
	Up	33945	3.87
4	Down	33270	3.73
	Up	32284	3.73
5	Down	32066	3.47
	Up	33563	3.33
Best Cost	32053		
Degree	3.6		
Time	6782.2		
Narrowed Sweep			
Sweep	**Pare**	**Best Cost**	**Degree**
1	Down	33061	3.47
	Up	34866	3.47
2	Down	35538	3.73
	Up	32566	3.6
3	Down	31388	3.47
	Up	32046	3.73
4	Down	31423	3.47
	Up	32281	3.73
5	Down	32185	3.6
	Up	32007	3.33
6	Down	31864	3.47
	Up	32898	3.47
7	Down	32683	3.2
	Up	32079	3.6
8	Down	31213	3.07
	Up	31147	3.07
9	Down	31717	3.6
	Up	33696	3.6
10	Down	32365	3.2
	Up	32666	3.33
Best Cost	31147		
Degree	3.07		
Time	6352.9		

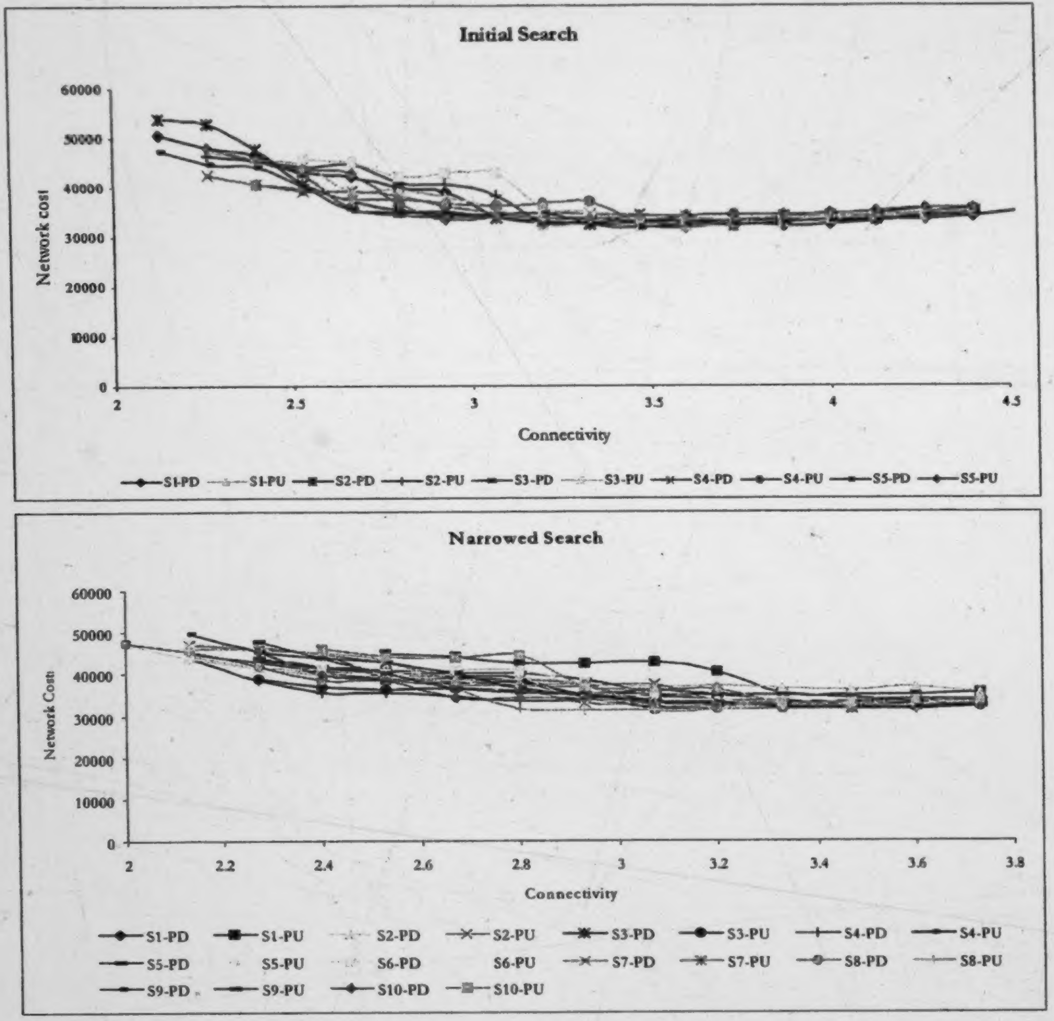


Figure 57 – Sweep Searching for network 15n59s

From Table 21 shows that the best found value for the initial sweep period was found in the 1st sweep. An improvement for this solution was only found in the 8th sweep within the narrowed search window.

Given the typically long time taken for 5 initial sweeps across the whole topology space, we could decided to reduce the number of sweeps in the initial search to concentrate on searching more within the narrowed region.

In comparison, the Iterative Sampling heuristic performed better on this same problem (finding a graph of cost of 30407 in a time of 3273.93 seconds). Recall that for this network, the MTRS-Reduced Edge set solution had a cost of 30743, obtained in a four-hour time-limited run. However, the Sweep Search best solution from the initial search has $\Delta = 5.4\%$ compared to the Iterative Sampling heuristic and this solution was obtained in a significantly less amount of time. Also, it is worthy to point out that the performance of the Sweep Search heuristic (convergence on a near-optimal solution) depends on the criterion used for adding or removing edges in the pare-up and pare down processes.

7.3 A COMPARISON OF OUR HEURISTIC RESULTS WITH 3-STEP HEURISTIC RESULTS

In some of our tests, the prior 3-step heuristic did not yield the optimal network design, but we see that it usually has an advantage in the speed of execution over other methods presented in this work for solving the complete MTRS problem. In this section, we attempt to improve on its performance. From [GrDo01], we understood that the heuristic run-time is often dominated by step W1. Step W1 of the 3-step MIP heuristic is to identify the optimal edge set for routing only. We try to bypass this step by substituting the selection of edges obtained from the W1 process with the MST for the network, and then pass this to the solver for steps S2 and J3 of the heuristic. However, the resulting solution was worse than the solution obtained by running the heuristic with all its steps. We attribute this to the set of edges selected in step S2, which are a result of edges selected in step W1 (or the MST in this case). Thus the edges present in the J3 problem did not include the edges for an optimal topology (or at least, a 2-node connected topology).

The conclusion we draw is that the set of edges selected for step W1 affect the quality of the final solution and therefore, is an important step in the heuristic.

From the *Edge/Cap* ratios of the solutions we obtained, the network cost is clearly dominated by the capacity costs. This has the effect of making the edge-to-capacity cost ratio Ω even smaller (since $\Omega = F_{ij}/c_{ij}$). In our studies of the effect of Ω on network costs, we saw that for small values of Ω, the optimal network has a very high average nodal degree (almost fully connected) and the connectivity decreases as Ω increases. One would then expect that for solutions that are dominated by capacity costs, the nodal degree would be high. This is not the case, and we see our solution having average degrees of 2.5 to 4 (such as expect from real transport networks).

8. APPLICATION TO INCREMENTAL GROWTH PLANNING

So far, we have studied the "green-fields" problem, where no spans are yet assumed to exist. This may actually be the case for a service provider who is a fresh entrant into the business. In reality, we can more often expect that service providers usually have some legacy topology, either acquired from another transport network company (in the case of a take-over, merger or buy-out) or some right-of-way land acquired from a utility company. In general, we also want to consider network growth planning where a service provider simply wishes to extend the reach of their existing transport network by adding more nodes and the associated spans to the network, or add new spans because demand growth warrants it.

In all these cases, we have a legacy topology (the existing topology) and we now wish to consider how best to grow or evolve the network. Which of the new spans under consideration should be added?

In this section, we attempt to extend our understanding of the MTRS problem on green-fields scenarios to applications in network growth planning. We make use of the previously defined variables and define two new ones to formulate a MIP structure for the topology growth problem. We earlier defined E as the set of edges in the graph. Now we define E_{exist} to be the set of edges in the legacy topology, E_{new} to be the candidate set of new spans from which we choose the additional spans and $E = E_{exist} \cup E_{new}$.

A MIP formulation for this problem can be stated as follows (the AMPL model is in Appendix C):

$$\textbf{Inc}-\textbf{MTRS:}\quad \textit{Minimize} \sum_{ij \in E_{new}} c_{ij}.\left(w_{ij}+s_{ij}\right)+F_{ij}.\delta_{ij}+\sum_{ij \in E_{exist}} c_{ij}.\left(w_{ij}+s_{ij}\right) \qquad (7.1)$$

Subject to:

1. For each demand, the total source flow equals the demand

$$\sum_{nj\in E} w_{nj}^{r} = d^{r} \qquad \forall r\in D, n = O[r] \qquad (7.2)$$

2. For each demand, the total sink flow equals the demand

$$\sum_{jn\in E} w_{jn}^{r} = d^{r} \qquad \forall r\in D, n = T[r] \qquad (7.3)$$

3. For each demand, the net flow (incoming – outgoing) for transshipment nodes is zero

$$\sum_{in\in E} w_{in}^{r} - \sum_{nj\in E} w_{nj}^{r} = 0 \qquad \forall r\in D, n\notin \{O[r], T[r]\} \qquad (7.4)$$

4. Working capacity on an edge must support the simultaneous flows over the edge

$$w_{ij} = \sum_{r\in D} w_{ij}^{r} \qquad \forall ij\in E \qquad (7.5)$$

5. For each span failure, the total source restoration flow equals the total affected working capacity

$$\sum_{ik\in E; j\neq k} s_{ij}^{ik} = w_{ij} \qquad \forall ij\in E \qquad (7.6)$$

6. For each span failure, the total sink restoration flow equals the total affected working capacity

$$\sum_{kj\in E; i\neq k} s_{ij}^{kj} = w_{ij} \qquad \forall ij\in E \qquad (7.7)$$

7. For each span failure, the net restoration flow for transshipment nodes is zero

$$\sum_{nk\in E; k\notin\{i,j\}} s_{ij}^{nk} - \sum_{kn\in E; k\notin\{i,j\}} s_{ij}^{kn} = 0 \qquad \forall ij\in E, \forall n\notin \{i, j\} \qquad (7.8)$$

8. Spare capacity on an edge is at least equal to the largest restoration flow over the edge

$$s_{kl} \geq s_{ij}^{kl}, s_{kl} \geq s_{ji}^{kl} \qquad \forall (ij), (kl)\in E^{2}, (ij)\neq (kl) \qquad (7.9)$$

9. The edge decision variables can only be 1 (edge included) or 0 (edge absent)

$$w_{ij} + s_{ij} \leq \delta_{ij}.K \qquad \delta_{ij} = \delta_{ji}, \delta_{ij} \in \{0,1\} \qquad w_{ij}, s_{ij} = \text{integer} \quad \forall ij\in E_{new} \qquad (7.10)$$

10. Each edge in the legacy topology is asserted (included) in the solution

$$\delta_{ij} = \delta_{ji} = 1 \qquad \forall ij \in E_{exist} \qquad (7.11)$$

11. Each node is at least of degree 2

$$\sum_{k \in N; i \neq k} \delta_{ik} \geq 2 \qquad \forall i \in N \qquad (7.12)$$

12. The network topology selected is at most of degree d_{max}.

$$\sum_{ij \in E_{new}} \delta_{ij} + E_{exist} \leq d_{max} \cdot N / 2 \qquad (7.13)$$

For the network growth problem, we consider the two network growth scenarios:

1. Extending the reach of the network by adding new nodes and the associated edges
2. Demand grows to a point that additional spans in the network become necessary between existing nodes.

For scenario 1, the set of demands we consider includes only the demands that involve the new nodes as either the origin or destination node. We implement this restriction to enable us control the problem size (constraints and variables). In principle, a more efficient or optimal design is obtained by considering the whole set of demands the network serves under the additional candidate edges. Such consideration could yield a more efficient routing and capacity design leading to a better network design.

For scenario 2, we allow the set of demands to include both the existing demands and the additional or new demands exchanged between O-D pairs (that is, not served by the existing network).

8.1 SCENARIO 1: EXTENDING THE REACH OF THE NETWORK

Knowing the complexity of the basic MTRS design problem, we used small incremental growth problem sizes for our tests. We studied the process of 'growing' the COST239-6n network to the COST239-9n network by adding 3 nodes which are not in the 9-node network. The candidate set of new spans includes all the 21 spans that the mere addition of 3 nodes contributes to the problem.

The 6-node network was obtained as the optimal solution to the MTRS problem for the demand quantities exchanged between its node pairs. In the case where demand grows between existing nodes, we consider that to be a capacity design problem on a fixed topology and therefore, not part of this problem. We only consider network growth due to topology change specifically, extending the reach and size of a network by adding new nodes. It is intuitive that the global optimization of the complete 9-node network given the complete demand set and all possible spans produces a better (optimal) solution than holding onto the legacy topology from the 6-node network and choosing additional new spans to add. We illustrate this growth process in the scenario described below.

Suppose we have a legacy topology with 6 nodes and 8 spans as shown in solid lines in Figure 58(a). We now wish to add 3 new nodes to our network. We assume the network to have just sufficient capacity to route the demands the network currently serves, and just

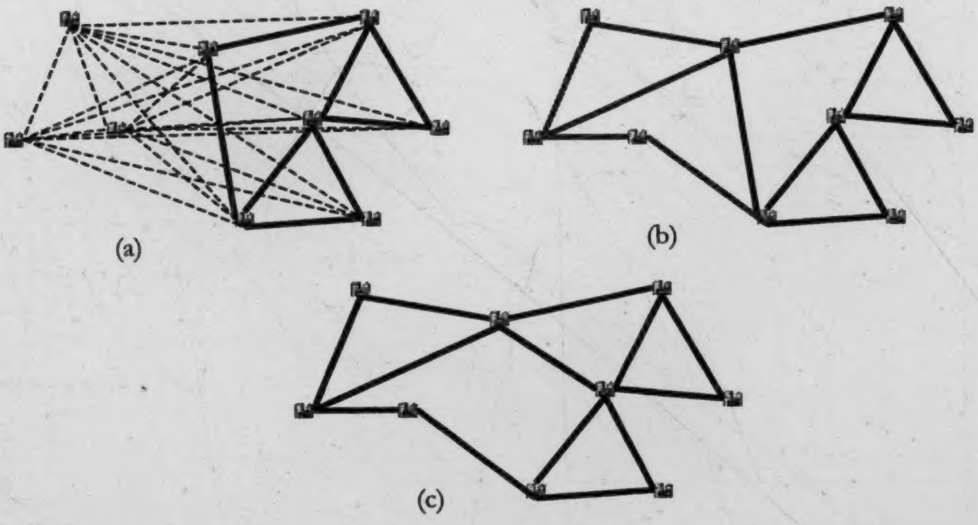


Figure 58 – The topology problem of growing a 6-node network to a 9-node network

sufficient spare required to restore traffic on any failed span. This means that the edges in the legacy topology come to the growth planning problem bearing a zero fixed cost and have zero capacity that can be used for the new demands, but new capacity can be added on them to route demands and restore failed traffic. If we were to consider the routing of all demands in the network, we may be able to reduce the amount of capacity (working and/or spare) on some spans in the legacy network.

Adding 3 new nodes introduces 21 possible spans to the problem indicated with broken lines in Figure 58a. Our task is to select which new spans to add to minimize total additional cost while serving all demands to the new nodes and retaining survivability as well. Figure 58b shows the solution with the topology growth MIP formulation. Figure 58c shows the solution to the COST239-9n MTRS problem (given the universe of edge candidates). The only topological difference is that one span that in the legacy 6-node network (and therefore forced into the solution for the growth problem) is not chosen as part of the optimal topology for the 9-node network when the solvers considers all candidate edges (another span that was not considered a good investment when considering the optimal 6-node network was optimal for the 9-node network).

The run-time for this problem (21 candidate edges and 21 demands) was 132.55 seconds whereas the solving for the optimal COST239-6n network had a run-time of 13.16 seconds (15 candidate edges and 15 demands). In contrast, solving the full 9-node problem has a solution time of 333451.09 seconds. The full problem gave a solution of cost 299673 whereas the new growth problem gives a solution with the total additional network cost of 151808, bringing the total cost of the network (existing edges and capacity plus additional edges and capacity) to a cost a 333992. This is about 11.5% higher than the solution for the full problem, but the run-time is less than 0.05% of the time needed to obtain the optimal solution.

The question we try to answer is: having solved a green fields problem of 11 nodes, how easily does this lend itself to solving the problem of extending the reach of say, a 30-node network by 11 new nodes?

The 11-node green-fields problem has a universe of 55 possible edges. We have seen how intractable this problem size proved to solve with the MIP solver. For a 30-node network, the addition of 11 new nodes presents a set of edge candidates to add that has a size of 385 spans. This does not include any spans that already exist in the network. In solving this problem, we have to include the existing edges (their presence in the final solution is enforced) because the MIP solver has to take advantage of edges already present for the capacity design.

Working and spare capacity will be placed on these old/existing edges for the routing of new demands to be served by the growth nodes and to ensure network survivability. This can only result in an increase in the problem size and thus, the complexity of the problem.

We find this problem to be as computationally complex as the green fields problem. The *N*-node incremental growth problem is not computationally equivalent to the *N*-node green-fields problem because we cannot separate the new growth nodes from the existing nodes due to the necessary interaction between the nodes and spans of the network. The interaction comes in the form of:

a. Paths connecting the new nodes with the existing nodes (causing the explosion in the edge set)

b. Demands exchanged between the new nodes and existing nodes, potentially the reason for growth in the first place.

If N is the number of nodes in an existing network and X is the number of new nodes, we can conclude that:

1. First, the X-node green-fields problem is not equivalent to the X-node incremental growth problem because the incremental problem includes a larger universe of candidate edges than the corresponding green-fields problem and this causes an explosion in the solution space.
2. Secondly, solving the X-node growth problem is computationally less intensive than the $(N+X)$-node green-fields network problem. If demand re-arrangeability is allowed, we can obtain solutions close to the optimal solution.

We have only considered the case where the model we are working with assumes that re-arrangeability of demands is not allowed. This means that demands the network currently serves cannot be re-routed. Suppose we remove this restriction, how does the solution change? We have the same network topology as the solution but because the routing of all demands is considered, there is an increased co-ordination of working and restoration routes that leads to reduced network costs. The solution now has a total cost of 306428 – only 2.25% worse than the solution for the complete problem and the solution time is 126.48 seconds (compare this with 333451.09 seconds required to obtain a solution for the complete problem).

If we apply the edge limiting heuristic, we achieve an average solution time of solution time of 3.03 seconds when we allow demand re-arrangeability, and a time of 3.28 seconds if this is not permitted. The solution costs remain the same as before for both instances. It is rather surprising that consistently, the solution time when demand re-arrangeability is allowed is smaller than when the reverse is the case. This is counter-intuitive, as one would expect that with fewer O-D pairs to consider, the problem would be easier to solve and therefore faster.

8.2 SCENARIO 2: INCREASING THE CONNECTIVITY OF THE NETWORK

In this scenario, we study the problem of growth in a network for which current demands have grown to a point where installing additional spans become more economical than mere capacity placement. The question then is what edges from the set of new edge candidate should be installed to minimize the total additional cost for the network?
We make one assumption: that the legacy topology is at least two-node connected. This assumption allows us to let go of constraint (7.12) for the MIP formulation. Our objective is simply to reduce the number of constraints in the problem.

The legacy topology we use (E_{exist}) is obtained as the optimal solution chosen from a given set of plausible of edges $E_{candidate}$. By plausible, we mean edges that connect geographically adjacent nodes as we would expect to see in a real transport network. Edges that were not selected as part of this legacy topology $(E_{candidate} \cap E_{exist})$ then form the set of candidate edges E_{new} for consideration in the growth problem. This choice is valid is we assume that the set of edges $E_{candidate}$ constitute the set of right-of-way contracts we are able to acquire, or represent feasible paths for installing the fiber cable. In this manner, we do not consider the complete problem where all $(N(N-1)/2)$ edges are included in the candidate edge set.

With the optimal topology we obtained as the legacy topology (E_{exist}), we try to solve the problem of increasing the connectivity of the network to minimize total additional costs. We have 2 approaches to solving this problem:

1. The set of candidate edges contains all edges of E_{new}. We solve the problem as a restricted instance of the MTRS problem and asserting all legacy topology spans in the solution. This would give the truly optimal solution to this problem. For small network sizes, it is the best approach. However, when considerably larger network sizes are considered or the edge set E_{new} is large, the use of a heuristic may be preferable to obtain solutions fast.
2. Using the Iterative Sampling technique, we add the spans one (or two) at a time to the legacy topology and evaluate the cost of the network. There are two ways to evaluate the cost of the network:
    a. Suppose the additional network cost incurred if the legacy topology serves and protects all demands without any span addition is C. For each new proposed topology, we can solve a capacity design problem and obtain the total additional network cost C'. We choose to accept the solution if $C' < \alpha C$ where $\alpha \geq 1$. This increases the probability of avoiding the problem with the Iterative Sampling heuristic (rejecting edges at step (n) that may have proved optimal at step $(n+1)$ when another edge is included in the candidate edge set.
    b. We solve a restricted instance of the MTRS problem where the candidate edge set is the subset of E_{new} added to the legacy topology.

We study this problem with a 10-node network for which $|E_{candidate}| = 24$. The demand matrix was generated using a random pattern for which every O-D pair had a non-zero value in the range [1, 5]. A solution to the MTRS problem for this (run-time = 5864.5 seconds) gave a solution with $|E_{exist}| = 13$ and $|E_{new}| = 11$. The total cost of this solution was obtained as 92820.6 with a topology having an average nodal degree of 2.6.

Another randomly generated demand pattern was used to simulate demand growth with each O-D pair having a value in the range [5, 10]. Any O-D pair for which there was no difference in its demand quantity represents one that did not grow. What would be the cost if the current network topology served the increased demands? We solve the capacity design problem using the JCP capacity design method. The total network cost was obtained as 217102.9. Can we minimize this cost by the addition of new spans?

Next, we solve a reference solution in which the optimal topology for serving the increased demand pattern is obtained with the candidate edge set of edges $E_{candidate}$. The solution is a network of cost 179054.6 with a topology of average nodal degree 3.6 (18 out of 24 edges were used). The run-time was 2723.56 seconds. Using the Iterative Sampling heuristic, we solve the growth-planning problem.

First, we solve the MTRS problem with the candidate edge set as all the new edges we are considering (i.e. $E_{candidate} = E_{new}$). The optimal solution found had a total network cost of 179323.7 and the average nodal degree was 3.8 (6 new edges added). The total run-time was 183.9 seconds.

Next, we add the edges one at a time and solving the MTRS problem with the candidate edge set consisting of all edges not part of the legacy topology. This means that for each iteration step, the edge candidate is made up the both the edges introduced in that step, and any edges which had been included in prior steps as part of the optimal solution. This means that any edge accepted as part of the solution in step(n) is still tested in step$(n+1)$ against the new edges included in step$(n+1)$. The total solution cost we obtain is 179323.7 with average nodal degree 3.8 in a run-time of 128.1 seconds.

Finally, we add the spans two at a time and solving the MTRS problem with the candidate edge made up as in the previous case. The total solution cost we obtain is 179323.7 with average nodal degree 3.8 in a run-time of 108.6 seconds.

In conclusion, we can say that where the network size is considered small and the number of new candidate edges being considered is also small, solving the MTRS problem with the candidate edge set as the set of all new edges is appropriate. For larger network sizes (or with larger number of candidate edges), using a heuristic such as the Iterative Sampling or Sweep Search heuristic would be of great benefit in reducing solution time because the reported solution times can only increase exponentially with the size of the edge candidate set. When many edges are included in each problem, the CPLEX run-time is greatly increased. Adding the span one or two at a time is only an effort to curtail the size of the edge candidate in each run of the MTRS problem. For this problem, adding the edges two at a time gave a better run-time overall, however, adding the edges one at a time solves each individual run faster.

9. CONCLUSION

9.1 SUMMARY OF THESIS

The complete Mesh Topology, Routing and Sparing (MTRS) problem is NP-hard and therefore justifies the pursuit of heuristics that give good approximating solutions within a reasonable time. An important issue for a heuristic aimed at solving the MTRS problem is the size of the network on which it can solve in a reasonable time frame. More specifically, it is the size of the candidate edge set in a topology and capacity optimization problem that strongly dominates its complexity.

The MIP solver for the MTRS problem on an unrestricted set of candidate edges is often bogged down in regions of the solution space containing high edge count because solutions of this nature are combinatorially more predominant. Our general hypothesis in this thesis work was that the solver could be helped to escape these regions by schemes that concentrate its search on feasible graphs having low edge count. We have contributed to studies on this type of problem by the heuristics developed and presented in this work. These are in summary:

1. Graph generation and sifting
2. A priori edge limitation
3. Sweep Search
4. Iterative Sampling

Let us say a few closing words about each.

In Chapter 4, we developed a graph generator and a graph sifter suite. The graph generator enumerated edge vectors which are evaluated by the sifter for certain properties which are important in a qualified transport network. The idea was to remove the burden of topology search from the MIP solver by having a fast "search engine" sift through topologies. Test results from this method, however, showed that the probability of getting a bi-connected graph through random generation and sifting was small and reduced with increase in problem size. Therefore, the use of algorithmic search methods is necessary.

Chapter 6 presents the three heuristics developed in this work. The first heuristic is the a priori limitation of candidate edge sets. When the whole universe of candidate edges is included in the problem, many generated graphs do not have the visual appearance of real transport networks because of the presence of edges that lie across the plane in Euclidean space. Our study of real networks shows that the nodes in these networks tend to connect to near neighbors. This led to the limitation heuristic.

The first step towards obtaining a solution to this problem is to limit the number of edges that are considered. The a priori edge limiting heuristic developed in this work does that quite effectively. It has been developed such that it is responsive to the inherent properties of the network and leaves an edge set of which the optimal edge set is a complete subset. From our results, we do not compromise the quality of the solution.

Given a universe of candidate edges, the heuristic tries to weed out edges which would not be included as part of the optimal MTRS solution. This is an attempt to limit the solution space where a qualified topology can be found. We studied typical optimal solutions to understand why certain edges are selected and others are rejected. Based on the empirical observations we had with these solutions, we developed the heuristic to mimic the decisions of the solver. Only edges that are plausible and have a high probability of being included in the solution are retained and other edges are removed.

From our experience with solving the MTRS problem, practical solutions can only be obtained by dividing the main problem into sub-instances of itself as used in the Iterative

Sampling or using a sequential approach in separating the components of the full problem (topology search and capacity design) as used for the Sweep Search

The second heuristic we developed is the Sweep Search heuristic. This heuristic divides the MTRS problem into two parts: Topology generation and Routing & Capacity design. Starting from a highly-connected graph, we vary the degree of connectivity and obtain a corresponding cost of edge establishment plus routing costs. After a number of sweeps through the topology space, we are able to arrive at a global or local minimum, which offers a good approximating solution.

The third heuristic is the iterative Sampling heuristic. The Iterative Sampling heuristic tries to solve the MTRS problem using the formal MIP solver. However, it tries to limit the solution space by solving multiple instances of the problem albeit with a limited edge set in each problem instance. Starting with a qualified topology, we solve the MTRS problem. Each edge that was not included in this first instance of the problem is added to the problem and evaluated against the edge set that the MIP solver found to be an optimal combination.

The Iterative Sampling heuristic does rather well for small network sizes but the performance degrades exponentially as the network size increases. The Sweep Search heuristic will always offer a solution that is close to optimal, which can be used as an upper bound for the MIP solver. However, the experience of the authors in [GrDo01] on providing upper bounds to the MIP solver show that setting such bounds inhibit the ability of the solver to find a feasible solution. This is because the solver is unable to find solutions that improve on the bound because it is still searching in a high connectivity topology space. This suggests that finding a closed connected graph on relatively few edges is the most difficult combinatorial aspect of the complete problem.

The conclusion we can draw from our studies is that applying the edge limiting heuristic followed by solving the MTRS problem for the graph is the first best choice. When this fails, we can use the Iterative sampling heuristic if the problem size is sufficiently small. For larger problem sizes, the Sweep Search heuristic would offer a better performance.

In this work, we used a specific demand intensity matrix for each design problem. Suppose the demand is uncertain or there is a measure of error in the given forecast, how would this affect the problem we solve and our approach to it? Our approach to solving the MTRS problem would not change. However, since our methods only allow the use of a specific demand data set, we then propose that we attempt to solve the problem using different demand data sets taken from within the region of uncertainty around the given demand forecast. We may find that the optimal solution over the range of demand scenarios is a robust or stable architecture as we found in Chapter 5.4.

An alternative approach would be to obtain a specific demand data as the normalized demand data set over different demand forecast scenarios. Such a normalized demand data set can have the value of each demand element as the mean over all demand scenarios for the particular O-D pair or some similar function. Using the normalized demand set, we can hopefully obtain a stable design solution over different demand scenarios.

9.2 DIRECTIONS FOR FURTHER STUDIES

The MTRS problem can be best solved using a divide-and-conquer approach – divide the problem into smaller instances, solve each instance and combine the solutions. This would have a similar structure to the topology growth problem and thus, our observation with the performance of the solver on this problem presents a new direction for further studies on heuristic approaches to the MTRS problem. This idea is developed next.

The complete topology can be broken into domains, similar to autonomous routing domains. The number of nodes in each domain (the domain size) would be some number that the CPLEX solver could solve easily, typically ≤ 7. Each domain is one sub-problem for which the MTRS problem is solved. The graph solutions for each instance can then be combined and a final MTRS problem can be solved linking the regions. The advantage would be the speed of execution. Each domain can be solved quickly because there are few nodes, the final solution would also be solved quickly because the edge set would only consist of very optimal edges and have a low edge count. Domain zoning principles could be based on:

1. Geography (spatial) – nodes that are geographically near would belong to a domain. This is in keeping with our observation that in feasible transport networks, nodes tend to connect to geographically near neighbors.
2. Demand – nodes that exchange high quantities of demand could be in a domain so that edges that are selected are optimized for serving the volume of traffic.

Another direction for further study would be the use of planarity as a sifting criterion. The use of this new criterion was suggested by Dr. M. MacGregor at the thesis defense for this work and extends from our belief that in feasible transport networks, nodes tend to connect to geographically near neighbors. This suggests the idea of planar graphs as qualified topologies for transport networks.

A planar graph is one that can be drawn on a plane piece of paper without any edge crossings – that is, so that no two edges intersect geometrically except at the vertex to which both are adjacent [Wils96]. There are well-known linear time algorithms for testing planarity in a network [TMNS90]. However, [Wils96] lists some basic theorems and corollaries of non-planar graphs:

1. If G is a connected simple planar graph with $n(\geq 3)$ vertices and m edges, then $m \leq 3n-6$.
2. If in addition, G has no triangle, then $m \leq 2n-4$. A triangle is a cycle in G with a hop-count of 3.
3. Every simple planar graph G contains a vertex of degree at most 5.

The question now is: should we use planarity as a graph sifting criterion? We do not have any conclusive evidence that the min-cost topology solution would always be non-planar. Thus, if planarity is used as a sifting criterion, we eliminate all non-planar graphs and we run the risk of eliminating the part of the solution space where the optimal topology solution lies. Thus the concept of planarity as a sifting criterion could be investigated in an attempt to reduce the solution space.

The draw back is that the graph generation plus sifter approach proved to be ineffective when considering network sizes larger than seven nodes. The probability of randomly enumerating a qualified topology is minimal thus the addition of additional sifting criteria may not provide added value since the problem is with the graph generation phase.

REFERENCES

[AhMO93] Ahuja, R. K., Magnanti, T. L., Orlin, J. B., *Network Flows: Theory, Algorithms and Applications*, Prentice Hall, New Jersey, 1993.

[BaMW95] Balakrishnan, A., Magnanti, T.L., Wong, R.T., "A decomposition network planning expansion", Operations Research vol. 43 1995 pg. 58 – 76.

[BaVG00] Baase S., Van Gelder, A., *Computer Algorithms: Introduction to Design and Analysis*, 3rd Edition, Addison-Wesley Longman, 2000.

[Bert98] Bertsekas, D. P., *Network optimization: Continuous and Discrete Models*, Athena Scientific, Belmont, 1998.

[Bhan99] Bhandari R., *Survivable Networks: Algorithms for Diverse Routing*, Kluwer Academic Publishers, 1999.

[CGKW02] Chang, F., Godin, A., Katsuda, D., Weil, J., "Understanding the values of Mesh networking in the Metro", *Proc. NFOEC 18th Annual Conference*, September 15 – 19, 2002, pg. 13 – 24.

[ClGr02] Clouqueur, M., Grover, W. D., "Mesh-Restorable Networks with Complete Dual-Failure restorability and with Selectively Enhanced Dual-Failure Restorability properties", *SPIE Optical Networking and Communications Conference (OptiComm 2002)*, Boston, MA, vol. 4874, 2002.

[DoGr00] Doucette, J., Grover, W. D., "Influence of Modularity and Economy of Scale Effects on Design of Mesh-restorable DWDM Networks", *IEEE Journal on Selected Areas In Communications*, vol. 18, no. 10, October 2000.

[DoGr02a] Doucette, J., Grover, W. D., "Capacity Design Studies of Span-Restorable Mesh Transport Networks with Shared-Risk Link Group (SRLG) effects", *SPIE Optical Networking and Communications Conference (OptiComm 2002)*, Boston, MA, July – August 2002.

[DoGr02b] Doucette, J., Grover, W. D., "Maintenance Immune Design of Span-Restorable Mesh Networks", *Proc. NFOEC 18th Annual Conference*, September 15 – 19, 2002, pg. 2049 – 2061.

[GaTD89] Gavish, B., Trudeau, P., Dror, M., Gendreau, M., Mason, L., "Fiber Optic Circuit Network design under Reliability Constraints", *IEEE Journal on Selected Areas in Communications*, vol 7, no. 8, 1989, pg. 1181 – 1187.

[Gavi91] Gavish, B., "Topological Design of Telecommunications network – Local access design methods", *Annals of Operations Research*, vol. 33, 1991 pg. 17 – 71.

[GDCL02] Grover, W. D., Doucette J., Clouqueur, M., Leung, D., Stamatelakis, D., "New Options and Insights for Survivable Transport Networks", *IEEE Communications Magazine*, January 2002, pg. 34 – 41.

[Gosa02] Gosal, S., "Metro Network Topologies and Economics", *Proc. NFOEC 18th Annual Conference*, September 15 – 19, 2002, pg. 517 – 525.

[GrBV91] Grover, W. D., Bilodeau, T. D., Venables, B. D., "Near Optimal Spare Capacity Planning in a Mesh-Restorable Network", *Proc. Of IEEE Globecom*, pg. 2007 – 2012.

[GrDo01] Grover, W. D., J. Doucette, "Topology Optimization of Survivable Mesh-Based Transport Networks", *Annals of Operations Research*, 2001, vol. 106, pg. 79 – 125.

[GrDo02] Grover, W. D., Doucette, J., "Design of a Meta-Mesh of Chain Sub-networks: Enhancing the attractiveness of a Mesh Restorable WDM Networking on Low Connectivity Graphs", *IEEE Journal on Selected Areas in*

Communications, January 2002, vol. 20, no. 1, pg. 47 – 61.

[GrSt98] Grover, W. D., Stamatelakis, D., "Cycle-Oriented Distributed Pre-Configurations: ring-like Speed with Mesh-like Capacity for Self-Planning Network Restoration", *Proc. IEEE ICC 1998*, Atlanta, GA, pg. 537 – 543, June 1998.

[HeBU95] Herzberg, M. Bye, S., Utano, A., "The hop limit approach for spare capacity assignment in survivable network", *IEEE/ACM Transactions on Networking*, vol. 3, no. 6, 1995, pg. 775 – 784.

[Herz93] Herzberg, M., "A decomposition approach to assign spare channels in self-healing network", *Proc. of IEEE Globecom*, 1993, pg. 1601 – 1605

[Hrom01] Hromkovic, J., *Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation and Heuristics*, Springer-Verlag, Berlin, 2001.

[Hum02] Hum, C., "Metro Optical network: Scalability and Evolution", *Proc. NFOEC 18th Annual Conference*, September 15 – 19, 2002, pg. 189 – 193.

[Kauf67] Kaufmann, A., *Graphs, Dynamic Programming and Finite Games*, Academic Press Inc., New York, 1967.

[Kers93] Kershenbaum, A., Telecommunications Network Design algorithms, McGraw-Hill, New York NY, 1993

[LiLH02] Lima, C.R., Lee, C., Hwu, C., "Evolution Path of Metro Optical Switched Networks, *Proc. NFOEC 18th Annual Conference*, September 15 – 19, 2002, pg. 1881 – 1888.

[MaRe71] Maxwell, L. M., Reed, M. B., *The theory of Graphs: A basis for Network Theory*, Pergamon Press, New York, 1971.

[MeMB00] Medina, A., Matta, I., Byers, J., "On the origin of Power Laws in Internet Topologies", *ACM SIGCOMM Computer Communications Review*, April 2000, vol. 30, no.2

[Mino81] Minoux, M., "Optimum Synthesis of a Network with non-simultaneous multi-commodity flow requirements", Studies on Graphs and Discrete Programming, P. Hansed, ed., North Holland Publishing Company, 1981, pg. 269 – 277.

[MoMP90] Monma, C. L., Munson, B., Pulleyblank., W., "Minimum Weight Two-Connected Spanning Networks", *Mathematical Programming*, vol. 6, 1990, pg. 153 – 171.

[MoRu99] Montgomery, D. C., Runger, G. C., Applied Statistics and Probability for Engineers, 2nd Edition, John Wiley and Sons, NY, 1999.

[Mosh89] Monma, C. L., Shallcross, D. F., "Methods for Designing Communication Networks with Certain Two-Connected Survivability Constraints", *Operations Research*, vol. 37, No. 4, July – August 1989, pg. 531 – 541.

[Orr02] Orr, J., "The Layered Network: Allowing for Cost Effective Migration to Next Generation Networks", *Proc. NFOEC 18th Annual Conference*, September 15 – 19, 2002, pg. 1047 – 1052.

[PiDe00] Pickavet, M., Demeester, P., "A Zoom-In Approach to Design SDH Mesh Restorable Networks", *Journal of Heuristics*, vol. 6, pg. 107 – 130, 2000.

[PoDe97 Poppe, F., Demeester P., "The Design of SDH Mesh Restorable Networks: Problem Formulation and Algorithm", *5th International Conference on Telecommunications Systems: Modeling and Analysis*, March 20 – 23, pg. 88 – 97, 1997.

[PPLD97] Pickavet, M., Poppe, F., Luystermans, J., Demeester, P., "A Genetic Algorithm for solving the Capacitated Survivable Network Design problem", *5th International Conference on Telecommunications Systems: Modeling and Analysis*, March 20 – 23, pg. 71 – 76, 1997.

[RMRR01] Ravi, R., Marathe M.V., Ravi, S.S., Rosenkrants D. J., Hunt, H. B., "Approximations Algorithms for Degree-Constrained Minimum Cost Network Design problems", *Algorithmica*, vol. 31, pg. 58 – 78, 2001.

[Robe99] Robertazzi, T. G., Planning Telecommunications Network, IEEE Press, 1999, pg. 66 – 72.

[SeWe02] Severi, M., Wellbrock, G., "Migration from point-to-point and ring architectures to meshed networks in DWDM: Limitations and possible architectures", *Proc. NFOEC 18th Annual Conference*, September 15 – 19, 2002, pg. 2010 – 2017.

[Smit82] Smith, D., *Network Optimization Practice: A Computational Guide*, Ellis Horwood Limited, Chichester, 1982.

[StDa94] Stoer, M., Dahl, G., "A Polyhedral approach to multi-commodity survivable network design", *Numerische Mathematik*, vol. 68, 1994, pg. 149 – 167

[Suur97] Suurballe, J. W., "Disjoint Paths in a Network", *Networks*, 1997, vol. 4, pg. 125 – 145.

[Temp81] Temperley, H. N. V., *Graph Theory and Applications*, Ellis Horwood Limited, Chichester, 1981.

[TMNS90] Tinhofer, G., Mayr, E., Noltemeien, H., Syslo, M.M., Albrecht, R., (eds) Computational Graph Theory, Springer-Verlag/Wien, New York, 1990.

[VePo02] Vernon, A. J., Portier, J.D., "Protection of Optical Channels in all optical

Networks", *Proc. NFOEC 18th Annual Conference*, September 15 – 19, 2002, pg. 1695 – 1706.

[West01] West, D. B, *Introduction to Graph Theory*, 2nd edition, Prentice Hall, New Jersey, 2001.

[Wils96] Wilson, R. J., *Introduction to Graph Theory*, 4th edition, Longman, 1996.

[Wins94] Winston, W., *Operations research: Applications and Algorithms*, 3rd Edition, Duxbury Press, California, 1994.

[ZeCB96] Zegura E., K. Calvert, Bhattacharjee S., "How to model an Internetwork", *IEEE INFOCOM,* 1996, vol. 2, pg. 594 – 602.

[ZeCD97] Zegura, E. W., Calvert, K.L, Donahoo, M.J., "A Quantitative Comparison of Graph-Based models for Internet Topology", *IEEE/ACM Transactions on Networking*, 1997, vol. 5, no. 6, pg. 770 – 783.

APPENDIX A

NETWORK DATA

NETWORK 26n127s

set NODES:=

N1 N2 N3 N4 N5 N6 N7 N8 N9 N10 N11 N12 N13 N14 N15 N16 N17 N18 N19 N20 N21 N22 N23 N24 N25 N26

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N2	42	S35	N6	N8	54	S69	N12	N10	41
S2	N1	N3	41	S36	N6	N10	81	S70	N12	N13	36
S3	N1	N6	110	S37	N6	N11	58	S71	N12	N14	45
S4	N1	N7	102	S38	N6	N15	153	S72	N12	N17	117
S5	N1	N11	161	S39	N6	N19	63	S73	N12	N18	105
S6	N1	N25	41	S40	N6	N20	73	S74	N12	N20	112
S7	N2	N3	41	S41	N6	N22	148	S75	N12	N25	190
S8	N2	N4	41	S42	N6	N25	112	S76	N13	N14	54
S9	N2	N5	73	S43	N7	N14	134	S77	N13	N15	63
S10	N2	N7	86	S44	N7	N19	42	S78	N13	N16	32
S11	N2	N8	130	S45	N7	N20	41	S79	N13	N18	69
S12	N2	N14	220	S46	N7	N21	69	S80	N13	N24	181
S13	N2	N18	148	S47	N7	N24	123	S81	N14	N9	122
S14	N2	N21	115	S48	N8	N9	41	S82	N14	N16	61
S15	N2	N23	136	S49	N8	N10	42	S83	N14	N17	110
S16	N2	N24	112	S50	N8	N11	50	S84	N14	N25	214
S17	N3	N6	73	S51	N8	N19	85	S85	N15	N12	81
S18	N3	N7	61	S52	N8	N20	114	S86	N15	N14	50
S19	N3	N13	139	S53	N8	N23	202	S87	N15	N16	42
S20	N3	N19	98	S54	N9	N11	72	S88	N15	N17	70
S21	N3	N26	50	S55	N9	N16	136	S89	N15	N19	92
S22	N4	N5	42	S56	N9	N23	239	S90	N15	N22	150
S23	N4	N6	41	S57	N10	N9	41	S91	N15	N25	197
S24	N4	N8	91	S58	N10	N11	36	S92	N16	N17	53
S25	N4	N12	143	S59	N10	N13	63	S93	N16	N19	50
S26	N4	N18	122	S60	N10	N22	192	S94	N16	N20	76
S27	N4	N22	143	S61	N11	N12	45	S95	N17	N18	32
S28	N4	N25	85	S62	N11	N13	41	S96	N17	N21	71
S29	N5	N8	63	S63	N11	N17	112	S97	N17	N22	80
S30	N5	N9	100	S64	N11	N18	90	S98	N18	N15	84
S31	N5	N10	103	S65	N11	N20	81	S99	N18	N16	50
S32	N5	N22	180	S66	N11	N23	172	S100	N18	N19	49
S33	N6	N5	41	S67	N11	N26	126	S101	N18	N20	40
S34	N6	N7	32	S68	N12	N9	80	S102	N18	N21	47

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S103	N18	N22	74	S112	N21	N22	47	S121	N23	N26	54
S104	N18	N23	100	S113	N21	N23	60	S122	N24	N1	76
S105	N18	N24	122	S114	N21	N25	68	S123	N24	N25	40
S106	N19	N13	41	S115	N21	N26	32	S124	N24	N26	52
S107	N19	N20	41	S116	N22	N23	34	S125	N25	N2	73
S108	N19	N24	142	S117	N22	N24	67	S126	N25	N7	92
S109	N20	N21	32	S118	N22	N26	60	S127	N25	N26	36
S110	N20	N22	78	S119	N23	N24	36				
S111	N20	N26	50	S120	N23	N25	65				

NETWORK 23n104s

set NODES:=

N1 N2 N3 N4 N5 N6 N7 N8 N9 N10 N11 N12 N13 N14 N15 N16 N17 N18 N19 N20 N21 N22 N23

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N14	197	S24	N4	N23	42.4	S47	N7	N8	42.4
S2	N1	N16	180.3	S25	N4	N3	44.7	S48	N7	N9	49.2
S3	N1	N17	164	S26	N4	N5	36.1	S49	N8	N15	40
S4	N1	N2	42.4	S27	N4	N6	72.8	S50	N8	N18	80.6
S5	N1	N20	108.2	S28	N4	N7	121.7	S51	N8	N23	121.7
S6	N1	N22	70.7	S29	N5	N11	139.3	S52	N8	N9	75.7
S7	N1	N3	89.4	S30	N5	N12	161.2	S53	N9	N10	39.1
S8	N1	N4	60	S31	N5	N19	30	S54	N9	N11	70.2
S9	N2	N17	174.6	S32	N5	N23	60.8	S55	N9	N20	187.9
S10	N2	N20	123.7	S33	N5	N6	56.6	S56	N9	N22	226.1
S11	N2	N23	60	S34	N5	N8	60.8	S57	N10	N11	44.7
S12	N2	N3	51	S35	N5	N9	136.5	S58	N10	N12	76.2
S13	N2	N4	42.4	S36	N6	N15	72.8	S59	N10	N13	90
S14	N3	N12	205.2	S37	N6	N19	80.6	S60	N10	N14	53.9
S15	N3	N13	200	S38	N6	N21	164	S61	N10	N15	53.9
S16	N3	N15	114	S39	N6	N7	50	S62	N10	N23	170
S17	N3	N21	158.1	S40	N6	N8	36.1	S63	N11	N12	31.6
S18	N3	N6	53.9	S41	N7	N12	130	S64	N11	N13	53.9
S19	N3	N7	102	S42	N7	N13	141.4	S65	N11	N14	41.2
S20	N4	N11	174.6	S43	N7	N14	100	S66	N12	N13	36.1
S21	N4	N14	144.2	S44	N7	N18	122.1	S67	N12	N14	53.9
S22	N4	N18	94.3	S45	N7	N19	114	S68	N12	N15	94.3
S23	N4	N19	58.3	S46	N7	N20	162.8	S69	N12	N16	80

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S70	N12	N18	121.7	S82	N15	N18	50	S94	N18	N22	100.5
S71	N12	N23	211.9	S83	N15	N19	60	S95	N19	N22	82.5
S72	N12	N9	100.1	S84	N15	N20	98.5	S96	N20	N21	36.1
S73	N13	N16	53.9	S85	N16	N15	50	S97	N20	N22	53.9
S74	N13	N17	100.5	S86	N16	N17	58.3	S98	N21	N16	120.4
S75	N13	N21	170.3	S87	N16	N18	44.7	S99	N21	N17	72.8
S76	N13	N22	196.5	S88	N17	N18	51	S100	N21	N22	44.7
S77	N13	N23	192.4	S89	N17	N20	56.6	S101	N21	N23	80
S78	N14	N13	44.7	S90	N17	N22	108.2	S102	N22	N23	44.7
S79	N14	N15	42.4	S91	N18	N19	36.1	S103	N23	N1	42.4
S80	N14	N16	36.1	S92	N18	N20	51	S104	N23	N20	67.1
S81	N14	N18	70	S93	N18	N21	85.4				

NETWORK 20n88s

set NODES:=

N1 N2 N3 N4 N5 N6 N7 N8 N9 N10 N11 N12 N13 N14 N15 N16 N17 N18 N19 N20

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N13	198	S21	N2	N7	140	S41	N6	N17	130
S2	N1	N14	149	S22	N3	N12	166	S42	N6	N7	36
S3	N1	N15	166	S23	N3	N18	45	S43	N6	N9	85
S4	N1	N18	95	S24	N3	N19	94	S44	N7	N10	70
S5	N1	N19	130	S25	N3	N5	50	S45	N7	N15	139
S6	N1	N2	36	S26	N4	N12	186	S46	N7	N16	177
S7	N1	N20	71	S27	N4	N15	150	S47	N7	N18	148
S8	N1	N4	81	S28	N4	N16	165	S48	N7	N20	177
S9	N1	N5	110	S29	N4	N6	51	S49	N7	N8	51
S10	N1	N7	161	S30	N4	N8	132	S50	N7	N9	50
S11	N2	N11	225	S31	N5	N10	75	S51	N8	N12	132
S12	N2	N13	163	S32	N5	N14	56	S52	N8	N13	148
S13	N2	N16	122	S33	N5	N18	83	S53	N8	N17	180
S14	N2	N17	89	S34	N5	N19	131	S54	N8	N9	54
S15	N2	N18	61	S35	N5	N6	54	S55	N9	N11	64
S16	N2	N20	50	S36	N5	N8	115	S56	N9	N12	80
S17	N2	N3	36	S37	N5	N9	91	S57	N9	N13	94
S18	N2	N4	72	S38	N6	N10	95	S58	N9	N17	139
S19	N2	N5	81	S39	N6	N13	140	S59	N9	N19	203
S20	N2	N6	114	S40	N6	N14	102	S60	N9	N20	200

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S61	N10	N11	73	S71	N12	N13	50	S81	N16	N18	61
S62	N10	N12	58	S72	N13	N14	50	S82	N16	N19	45
S63	N10	N13	54	S73	N13	N15	50	S83	N17	N18	36
S64	N10	N14	51	S74	N13	N16	101	S84	N18	N19	50
S65	N10	N17	102	S75	N13	N18	121	S85	N18	N20	45
S66	N10	N18	130	S76	N14	N15	45	S86	N19	N15	91
S67	N11	N14	120	S77	N14	N16	86	S87	N19	N20	61
S68	N11	N15	143	S78	N14	N19	114	S88	N20	N16	94
S69	N11	N8	102	S79	N15	N16	51				
S70	N12	N11	50	S80	N15	N17	42				

NETWORK 15n59s

set NODES:=

N1 N2 N3 N4 N5 N6 N7 N8 N9 N10 N11 N12 N13 N14 N15

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N10	40	S21	N4	N7	19	S41	N7	N9	11
S2	N1	N15	22	S22	N4	N8	37	S42	N8	N11	56
S3	N1	N2	41	S23	N4	N9	28	S43	N8	N12	18
S4	N1	N3	10	S24	N5	N10	33	S44	N8	N9	29
S5	N1	N4	25	S25	N5	N12	18	S45	N9	N10	16
S6	N1	N7	31	S26	N5	N13	20	S46	N9	N11	28
S7	N2	N14	15	S27	N5	N6	30	S47	N9	N12	38
S8	N2	N3	34	S28	N5	N7	29	S48	N9	N13	25
S9	N2	N4	21	S29	N5	N8	18	S49	N9	N14	37.2
S10	N2	N8	41	S30	N5	N9	20	S50	N9	N15	26
S11	N2	N9	46	S31	N6	N12	33	S51	N10	N11	15
S12	N3	N10	34	S32	N6	N14	11	S52	N10	N13	28
S13	N3	N14	36	S33	N6	N7	32	S53	N10	N15	19
S14	N3	N15	19	S34	N6	N8	15	S54	N10	N4	37.5
S15	N3	N4	15	S35	N6	N9	32	S55	N11	N13	26
S16	N3	N7	22	S36	N7	N10	18	S56	N11	N15	31
S17	N4	N14	21	S37	N7	N11	34	S57	N11	N5	40.4
S18	N4	N15	29	S38	N7	N14	32	S58	N12	N13	34
S19	N4	N5	42	S39	N7	N15	19	S59	N14	N15	49
S20	N4	N6	26	S40	N7	N8	34				

NETWORK 11n55s1

set NODES:=

N1 N2 N3 N4 N5 N6 N7 N8 N9 N10 N11

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N10	63	S20	N3	N10	80	S39	N5	N8	63
S2	N1	N11	40	S21	N3	N11	67	S40	N5	N9	56
S3	N1	N2	22	S22	N3	N4	28	S41	N6	N10	50
S4	N1	N3	36	S23	N3	N5	41	S42	N6	N11	53
S5	N1	N4	50	S24	N3	N6	44	S43	N6	N7	30
S6	N1	N5	70	S25	N3	N7	64	S44	N6	N8	30
S7	N1	N6	53	S26	N3	N8	72	S45	N6	N9	22
S8	N1	N7	82	S27	N3	N9	50	S46	N7	N10	72
S9	N1	N8	70	S28	N4	N10	67	S47	N7	N11	82
S10	N1	N9	42	S29	N4	N11	64	S48	N7	N8	42
S11	N2	N10	50	S30	N4	N5	22	S49	N7	N9	50
S12	N2	N11	36	S31	N4	N6	20	S50	N8	N10	31
S13	N2	N3	31	S32	N4	N7	36	S51	N8	N11	50
S14	N2	N4	31	S33	N4	N8	50	S52	N8	N9	28
S15	N2	N5	53	S34	N4	N9	36	S53	N9	N10	31
S16	N2	N6	31	S35	N5	N10	86	S54	N9	N11	31
S17	N2	N7	60	S36	N5	N11	86	S55	N10	N11	28
S18	N2	N8	50	S37	N5	N6	36				
S19	N2	N9	22	S38	N5	N7	31				

NETWORK COST239-11nodes

set NODES:=

N0 N1 N2 N3 N4 N5 N6 N7 N8 N9 N10

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N0	N1	820	S7	N0	N7	400	S13	N1	N4	820
S2	N0	N2	600	S8	N0	N8	300	S14	N1	N5	966
S3	N0	N3	994	S9	N0	N9	450	S15	N1	N6	920
S4	N0	N4	1387	S10	N0	N10	1387	S16	N1	N7	678
S5	N0	N5	1090	S11	N1	N2	320	S17	N1	N8	930
S6	N0	N6	716	S12	N1	N3	549	S18	N1	N9	1255

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S19	N1	N10	1356	S32	N3	N8	779	S45	N5	N10	390
S20	N2	N3	565	S33	N3	N9	1106	S46	N6	N7	390
S21	N2	N4	730	S34	N3	N10	740	S47	N6	N8	210
S22	N2	N5	799	S35	N4	N5	660	S48	N6	N9	550
S23	N2	N6	685	S36	N4	N6	1140	S49	N6	N10	760
S24	N2	N7	350	S37	N4	N7	982	S50	N7	N8	220
S25	N2	N8	671	S38	N4	N8	1262	S51	N7	N9	635
S26	N2	N9	1055	S39	N4	N9	1616	S52	N7	N10	851
S27	N2	N10	1130	S40	N4	N10	1250	S53	N8	N9	390
S28	N3	N4	320	S41	N5	N6	660	S54	N8	N10	1009
S29	N3	N5	340	S42	N5	N7	788	S55	N9	N10	1310
S30	N3	N6	617	S43	N5	N8	1065				
S31	N3	N7	730	S44	N5	N9	1303				

NETWORK 10n45s1

set NODES:=

N1 N2 N3 N4 N5 N6 N7 N8 N9 N10

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N10	363.9	S16	N2	N8	511.4	S31	N5	N10	444.1
S2	N1	N2	241.8	S17	N2	N9	213.7	S32	N5	N6	355.6
S3	N1	N3	154.3	S18	N3	N10	252.2	S33	N5	N7	281.7
S4	N1	N4	176.3	S19	N3	N4	250.9	S34	N5	N8	481.8
S5	N1	N5	369	S20	N3	N5	443.9	S35	N5	N9	194.4
S6	N1	N6	359.2	S21	N3	N6	510.9	S36	N6	N10	663.1
S7	N1	N7	347.4	S22	N3	N7	309.1	S37	N6	N7	559.2
S8	N1	N8	282	S23	N3	N8	138.1	S38	N6	N8	623.7
S9	N1	N9	383.1	S24	N3	N9	509.6	S39	N6	N9	184.6
S10	N2	N10	561.5	S25	N4	N10	327.9	S40	N7	N10	175.2
S11	N2	N3	394.5	S26	N4	N5	198.3	S41	N7	N8	264.6
S12	N2	N4	248.2	S27	N4	N6	338.3	S42	N7	N9	451
S13	N2	N5	326.4	S28	N4	N7	228.8	S43	N8	N10	137.9
S14	N2	N6	117.8	S29	N4	N8	316.8	S44	N8	N9	590.3
S15	N2	N7	476.6	S30	N4	N9	273.7	S45	N9	N10	590.6

NETWORK 10n45s2

set NODES:=

N1 N2 N3 N4 N5 N6 N7 N8 N9 N10

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N10	22	S16	N2	N8	70	S31	N5	N10	28
S2	N1	N2	22	S17	N2	N9	72	S32	N5	N6	31
S3	N1	N3	50	S18	N3	N10	63	S33	N5	N7	22
S4	N1	N4	44	S19	N3	N4	22	S34	N5	N8	42
S5	N1	N5	30	S20	N3	N5	40	S35	N5	N9	50
S6	N1	N6	60	S21	N3	N6	42	S36	N6	N10	58
S7	N1	N7	44	S22	N3	N7	60	S37	N6	N7	36
S8	N1	N8	67	S23	N3	N8	76	S38	N6	N8	40
S9	N1	N9	58	S24	N3	N9	90	S39	N6	N9	67
S10	N2	N10	40	S25	N4	N10	50	S40	N7	N10	30
S11	N2	N3	28	S26	N4	N5	22	S41	N7	N8	22
S12	N2	N4	30	S27	N4	N6	22	S42	N7	N9	31
S13	N2	N5	28	S28	N4	N7	40	S43	N8	N10	50
S14	N2	N6	50	S29	N4	N8	53	S44	N8	N9	36
S15	N2	N7	50	S30	N4	N9	70	S45	N9	N10	36

NETWORK COST239-10nodes

set NODES:=

N0 N1 N2 N3 N4 N5 N6 N7 N8 N9

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N0	N1	820	S11	N1	N3	549	S21	N2	N6	685
S2	N0	N2	600	S12	N1	N4	820	S22	N2	N7	350
S3	N0	N3	994	S13	N1	N5	966	S23	N2	N8	671
S4	N0	N4	1387	S14	N1	N6	920	S24	N2	N9	1055
S5	N0	N5	1090	S15	N1	N7	678	S25	N3	N4	320
S6	N0	N6	716	S16	N1	N8	930	S26	N3	N5	340
S7	N0	N7	400	S17	N1	N9	1255	S27	N3	N6	617
S8	N0	N8	300	S18	N2	N3	565	S28	N3	N7	730
S9	N0	N9	450	S19	N2	N4	730	S29	N3	N8	779
S10	N1	N2	320	S20	N2	N5	799	S30	N3	N9	1106

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S31	N4	N5	660	S36	N5	N6	660	S41	N6	N8	210
S32	N4	N6	1140	S37	N5	N7	788	S42	N6	N9	550
S33	N4	N7	982	S38	N5	N8	1065	S43	N7	N8	220
S35	N4	N9	1616	S39	N5	N9	1303	S44	N7	N9	635
S34	N4	N8	1262	S40	N6	N7	390	S45	N8	N9	390

NETWORK COST239-9nodes

set NODES:=

N1 N2 N3 N4 N5 N6 N7 N8 N9

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N2	320	S13	N2	N7	350	S25	N4	N8	1262
S2	N1	N3	549	S14	N2	N8	671	S26	N4	N9	1616
S3	N1	N4	820	S15	N2	N9	1055	S27	N5	N6	660
S4	N1	N5	966	S16	N3	N4	320	S28	N5	N7	788
S5	N1	N6	920	S17	N3	N5	340	S29	N5	N8	1065
S6	N1	N7	678	S18	N3	N6	617	S30	N5	N9	1303
S7	N1	N8	930	S19	N3	N7	730	S31	N6	N7	390
S8	N1	N9	1255	S20	N3	N8	779	S32	N6	N8	210
S9	N2	N3	565	S21	N3	N9	1106	S33	N6	N9	550
S10	N2	N4	730	S22	N4	N5	660	S34	N7	N8	220
S11	N2	N5	799	S23	N4	N6	1140	S35	N7	N9	635
S12	N2	N6	685	S24	N4	N7	982	S36	N8	N9	390

NETWORK 9n36s1

set NODES:=

N1 N2 N3 N4 N5 N6 N7 N8 N9

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N2	36	S5	N1	N6	53	S9	N2	N3	31
S2	N1	N3	30	S6	N1	N7	80	S10	N2	N4	31
S3	N1	N4	53	S7	N1	N8	58	S11	N2	N5	58
S4	N1	N5	70	S8	N1	N9	31	S12	N2	N6	58

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S13	N2	N7	86	S21	N3	N9	36	S29	N5	N8	64
S14	N2	N8	80	S22	N4	N5	28	S30	N5	N9	67
S15	N2	N9	60	S23	N4	N6	40	S31	N6	N7	28
S16	N3	N4	28	S24	N4	N7	63	S32	N6	N8	36
S17	N3	N5	40	S25	N4	N8	72	S33	N6	N9	41
S18	N3	N6	28	S26	N4	N9	64	S34	N7	N8	41
S19	N3	N7	56	S27	N5	N6	28	S35	N7	N9	60
S20	N3	N8	50	S28	N5	N7	40	S36	N8	N9	28

NETWORK 9n36s2

set NODES:=

N1 N2 N3 N4 N5 N6 N7 N8 N9

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N2	36	S13	N2	N7	90	S25	N4	N8	64
S2	N1	N3	72	S14	N2	N8	82	S26	N4	N9	70
S3	N1	N4	63	S15	N2	N9	64	S27	N5	N6	28
S4	N1	N5	28	S16	N3	N4	41	S28	N5	N7	53
S5	N1	N6	56	S17	N3	N5	64	S29	N5	N8	41
S6	N1	N7	80	S18	N3	N6	67	S30	N5	N9	31
S7	N1	N8	60	S19	N3	N7	94	S31	N6	N7	30
S8	N1	N9	31	S20	N3	N8	100	S32	N6	N8	36
S9	N2	N3	40	S21	N3	N9	94	S33	N6	N9	50
S10	N2	N4	50	S22	N4	N5	40	S34	N7	N8	31
S11	N2	N5	41	S23	N4	N6	28	S35	N7	N9	64
S12	N2	N6	60	S24	N4	N7	53	S36	N8	N9	36

NETWORK COST239-8nodes

set NODES:=

N1 N2 N3 N4 N5 N6 N7 N8

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N2	320	S6	N1	N7	920	S11	N2	N6	685
S2	N1	N3	549	S7	N1	N8	678	S12	N2	N7	350
S3	N1	N4	820	S8	N2	N3	930	S13	N2	N8	671
S4	N1	N5	966	S9	N2	N4	730	S14	N3	N4	320
S5	N1	N6	920	S10	N2	N5	799	S15	N3	N5	340
S16	N3	N6	617	S21	N4	N7	982	S26	N6	N7	390
S17	N3	N7	730	S22	N4	N8	1262	S27	N6	N8	210
S18	N3	N8	779	S23	N5	N6	660	S28	N7	N8	220
S19	N4	N5	660	S24	N5	N7	788				
S20	N4	N6	1140	S25	N5	N8	1065				

NETWORK 8n28s1

set NODES:=

N1 N2 N3 N4 N5 N6 N7 N8

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N2	260.2	S11	N2	N6	366.7	S21	N4	N7	213.2
S2	N1	N3	286.1	S12	N2	N7	396.1	S22	N4	N8	461.5
S3	N1	N4	353	S13	N2	N8	467.1	S23	N5	N6	410.6
S4	N1	N5	325.7	S14	N3	N4	428.4	S24	N5	N7	284.1
S5	N1	N6	124.8	S15	N3	N5	191.7	S25	N5	N8	192.1
S6	N1	N7	146	S16	N3	N6	406.1	S26	N6	N7	146
S7	N1	N8	468	S17	N3	N7	335.3	S27	N6	N8	574
S8	N2	N3	261.1	S18	N3	N8	210.9	S28	N7	N8	464.5
S9	N2	N4	577.7	S19	N4	N5	275.5				
S10	N2	N5	426.2	S20	N4	N6	348.3				

NETWORK 8n28s2

set NODES:=

N1 N2 N3 N4 N5 N6 N7 N8

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N2	200.5	S11	N2	N6	412.2	S21	N4	N7	365.5
S2	N1	N3	144.5	S12	N2	N7	384.6	S22	N4	N8	167.2
S3	N1	N4	257.8	S13	N2	N8	386	S23	N5	N6	315.9
S4	N1	N5	164	S14	N3	N4	402	S24	N5	N7	317.1
S5	N1	N6	212.5	S15	N3	N5	165.1	S25	N5	N8	474.7
S6	N1	N7	185.3	S16	N3	N6	160.7	S26	N6	N7	64.2
S7	N1	N8	325.3	S17	N3	N7	182.7	S27	N6	N8	407.8
S8	N2	N3	304.8	S18	N3	N8	451.8	S28	N7	N8	343.8
S9	N2	N4	236.7	S19	N4	N5	367.6				
S10	N2	N5	182.1	S20	N4	N6	421.3				

NETWORK COST239-7nodes

set NODES:=

N1 N2 N3 N4 N5 N6 N7

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N2	320	S8	N2	N4	730	S15	N3	N7	730
S2	N1	N3	549	S9	N2	N5	799	S16	N4	N5	660
S3	N1	N4	820	S10	N2	N6	685	S17	N4	N6	1140
S4	N1	N5	966	S11	N2	N7	350	S18	N4	N7	982
S5	N1	N6	920	S12	N3	N4	320	S19	N5	N6	660
S6	N1	N7	678	S13	N3	N5	340	S20	N5	N7	788
S7	N2	N3	565	S14	N3	N6	617	S21	N6	N7	390

NETWORK 7n21s1

set NODES:=

N1 N2 N3 N4 N5 N6 N7

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N2	282.4	S8	N2	N4	357.1	S15	N3	N7	195.3
S2	N1	N3	396.1	S9	N2	N5	159	S16	N4	N5	508
S3	N1	N4	189.4	S10	N2	N6	424.6	S17	N4	N6	534
S4	N1	N5	439.9	S11	N2	N7	433.2	S18	N4	N7	231.8
S5	N1	N6	608.9	S12	N3	N4	304.7	S19	N5	N6	405.3
S6	N1	N7	400.8	S13	N3	N5	366.4	S20	N5	N7	541.4
S7	N2	N3	295.7	S14	N3	N6	229.3	S21	N6	N7	385.4

NETWORK 7n21s2

set NODES:=

N1 N2 N3 N4 N5 N6 N7

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N2	210.5	S8	N2	N4	500	S15	N3	N7	309.7
S2	N1	N3	394.9	S9	N2	N5	432	S16	N4	N5	307.4
S3	N1	N4	315.6	S10	N2	N6	291.7	S17	N4	N6	208.4
S4	N1	N5	373.9	S11	N2	N7	361.8	S18	N4	N7	177.9
S5	N1	N6	126.6	S12	N3	N4	485.3	S19	N5	N6	267
S6	N1	N7	233	S13	N3	N5	220.9	S20	N5	N7	169.5
S7	N2	N3	333.5	S14	N3	N6	349.3	S21	N6	N7	109.3

NETWORK COST239-6nodes

set NODES:=

N1 N2 N3 N4 N5 N6

set SPANS:=

Span	O	D	Length	Span	O	D	Length	Span	O	D	Length
S1	N1	N2	320	S6	N2	N3	565	S11	N3	N5	340
S2	N1	N3	549	S7	N2	N4	730	S12	N3	N6	617
S3	N1	N4	820	S8	N2	N5	799	S13	N4	N5	660
S4	N1	N5	966	S9	N2	N6	685	S14	N4	N6	1140
S5	N1	N6	920	S10	N3	N4	320	S15	N5	N6	660

APPENDIX B

DEMAND DATA

NETWORK 26n127s

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N10	1	D41	N10	N25	2	D81	N12	N20	4
D2	N1	N11	2	D42	N10	N26	2	D82	N12	N21	3
D3	N1	N12	2	D43	N10	N3	2	D83	N12	N22	3
D4	N1	N13	2	D44	N10	N4	2	D84	N12	N23	2
D5	N1	N14	1	D45	N10	N5	2	D85	N12	N24	2
D6	N1	N15	1	D46	N10	N6	4	D86	N12	N25	2
D7	N1	N16	1	D47	N10	N7	3	D87	N12	N26	2
D8	N1	N17	1	D48	N10	N8	6	D88	N12	N3	2
D9	N1	N18	3	D49	N10	N9	5	D89	N12	N4	2
D10	N1	N19	2	D50	N11	N12	9	D90	N12	N5	2
D11	N1	N2	6	D51	N11	N13	9	D91	N12	N6	5
D12	N1	N20	3	D52	N11	N14	4	D92	N12	N7	4
D13	N1	N21	2	D53	N11	N15	4	D93	N12	N8	5
D14	N1	N22	2	D54	N11	N16	5	D94	N12	N9	4
D15	N1	N23	2	D55	N11	N17	3	D95	N13	N14	6
D16	N1	N24	3	D56	N11	N18	6	D96	N13	N15	5
D17	N1	N25	6	D57	N11	N19	9	D97	N13	N16	8
D18	N1	N26	3	D58	N11	N2	4	D98	N13	N17	3
D19	N1	N3	4	D59	N11	N20	5	D99	N13	N18	7
D20	N1	N4	3	D60	N11	N21	3	D100	N13	N19	8
D21	N1	N5	2	D61	N11	N22	3	D101	N13	N2	3
D22	N1	N6	3	D62	N11	N23	2	D102	N13	N20	5
D23	N1	N7	3	D63	N11	N24	3	D103	N13	N21	3
D24	N1	N8	2	D64	N11	N25	3	D104	N13	N22	3
D25	N1	N9	1	D65	N11	N26	3	D105	N13	N23	2
D26	N10	N11	8	D66	N11	N3	3	D106	N13	N24	2
D27	N10	N12	7	D67	N11	N4	3	D107	N13	N25	3
D28	N10	N13	4	D68	N11	N5	3	D108	N13	N26	2
D29	N10	N14	3	D69	N11	N6	9	D109	N13	N3	2
D30	N10	N15	2	D70	N11	N7	6	D110	N13	N4	2
D31	N10	N16	3	D71	N11	N8	8	D111	N13	N5	2
D32	N10	N17	2	D72	N11	N9	4	D112	N13	N6	5
D33	N10	N18	3	D73	N12	N13	10	D113	N13	N7	4
D34	N10	N19	4	D74	N12	N14	7	D114	N13	N8	4
D35	N10	N2	2	D75	N12	N15	5	D115	N13	N9	3
D36	N10	N20	3	D76	N12	N16	5	D116	N14	N15	6
D37	N10	N21	2	D77	N12	N17	3	D117	N14	N16	4
D38	N10	N22	2	D78	N12	N18	5	D118	N14	N17	2
D39	N10	N23	2	D79	N12	N19	5	D119	N14	N18	4
D40	N10	N24	2	D80	N12	N2	3	D120	N14	N19	3

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D121	N14	N2	2	D163	N16	N24	2	D205	N18	N9	3
D122	N14	N20	3	D164	N16	N25	2	D206	N19	N2	3
D123	N14	N21	2	D165	N16	N26	2	D207	N19	N20	9
D124	N14	N22	2	D166	N16	N3	2	D208	N19	N21	4
D125	N14	N23	2	D167	N16	N4	2	D209	N19	N22	4
D126	N14	N24	2	D168	N16	N5	2	D210	N19	N23	3
D127	N14	N25	2	D169	N16	N6	3	D211	N19	N24	3
D128	N14	N26	2	D170	N16	N7	3	D212	N19	N25	3
D129	N14	N3	1	D171	N16	N8	2	D213	N19	N26	3
D130	N14	N4	2	D172	N16	N9	2	D214	N19	N3	3
D131	N14	N5	2	D173	N17	N18	11	D215	N19	N4	3
D132	N14	N6	3	D174	N17	N19	4	D216	N19	N5	3
D133	N14	N7	3	D175	N17	N2	2	D217	N19	N6	7
D134	N14	N8	3	D176	N17	N20	4	D218	N19	N7	8
D135	N14	N9	2	D177	N17	N21	4	D219	N19	N8	4
D136	N15	N16	6	D178	N17	N22	4	D220	N19	N9	3
D137	N15	N17	4	D179	N17	N23	2	D221	N2	N20	4
D138	N15	N18	6	D180	N17	N24	2	D222	N2	N21	3
D139	N15	N19	4	D181	N17	N25	2	D223	N2	N22	3
D140	N15	N2	2	D182	N17	N26	2	D224	N2	N23	3
D141	N15	N20	3	D183	N17	N3	2	D225	N2	N24	4
D142	N15	N21	3	D184	N17	N4	2	D226	N2	N25	6
D143	N15	N22	3	D185	N17	N5	1	D227	N2	N26	4
D144	N15	N23	2	D186	N17	N6	3	D228	N2	N3	7
D145	N15	N24	2	D187	N17	N7	3	D229	N2	N4	7
D146	N15	N25	2	D188	N17	N8	2	D230	N2	N5	4
D147	N15	N26	2	D189	N17	N9	2	D231	N2	N6	6
D148	N15	N3	2	D190	N18	N19	9	D232	N2	N7	5
D149	N15	N4	2	D191	N18	N2	4	D233	N2	N8	3
D150	N15	N5	2	D192	N18	N20	12	D234	N2	N9	2
D151	N15	N6	3	D193	N18	N21	8	D235	N20	N21	10
D152	N15	N7	3	D194	N18	N22	7	D236	N20	N22	6
D153	N15	N8	2	D195	N18	N23	4	D237	N20	N23	4
D154	N15	N9	2	D196	N18	N24	4	D238	N20	N24	4
D155	N16	N17	4	D197	N18	N25	5	D239	N20	N25	5
D156	N16	N18	7	D198	N18	N26	5	D240	N20	N26	6
D157	N16	N19	5	D199	N18	N3	3	D241	N20	N3	4
D158	N16	N2	2	D200	N18	N4	3	D242	N20	N4	4
D159	N16	N20	4	D201	N18	N5	3	D243	N20	N5	3
D160	N16	N21	3	D202	N18	N6	6	D244	N20	N6	6
D161	N16	N22	3	D203	N18	N7	6	D245	N20	N7	9
D162	N16	N23	2	D204	N18	N8	4	D246	N20	N8	3

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D247	N20	N9	2	D274	N23	N3	2	D301	N26	N6	4
D248	N21	N22	7	D275	N23	N4	2	D302	N26	N7	4
D249	N21	N23	5	D276	N23	N5	2	D303	N26	N8	2
D250	N21	N24	4	D277	N23	N6	3	D304	N26	N9	2
D251	N21	N25	5	D278	N23	N7	3	D305	N3	N4	4
D252	N21	N26	7	D279	N23	N8	2	D306	N3	N5	2
D253	N21	N3	3	D280	N23	N9	1	D307	N3	N6	4
D254	N21	N4	3	D281	N24	N25	9	D308	N3	N7	4
D255	N21	N5	2	D282	N24	N26	5	D309	N3	N8	2
D256	N21	N6	4	D283	N24	N3	3	D310	N3	N9	2
D257	N21	N7	4	D284	N24	N4	2	D311	N4	N5	4
D258	N21	N8	2	D285	N24	N5	2	D312	N4	N6	8
D259	N21	N9	2	D286	N24	N6	3	D313	N4	N7	5
D260	N22	N23	10	D287	N24	N7	3	D314	N4	N8	3
D261	N22	N24	6	D288	N24	N8	2	D315	N4	N9	2
D262	N22	N25	5	D289	N24	N9	2	D316	N5	N6	7
D263	N22	N26	5	D290	N25	N26	8	D317	N5	N7	3
D264	N22	N3	3	D291	N25	N3	6	D318	N5	N8	4
D265	N22	N4	3	D292	N25	N4	4	D319	N5	N9	2
D266	N22	N5	2	D293	N25	N5	2	D320	N6	N7	13
D267	N22	N6	4	D294	N25	N6	4	D321	N6	N8	8
D268	N22	N7	4	D295	N25	N7	4	D322	N6	N9	4
D269	N22	N8	2	D296	N25	N8	2	D323	N7	N8	4
D270	N22	N9	2	D297	N25	N9	2	D324	N7	N9	3
D271	N23	N24	8	D298	N26	N3	4	D325	N8	N9	6
D272	N23	N25	5	D299	N26	N4	3				
D273	N23	N26	4	D300	N26	N5	2				

NETWORK 23n104s

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	5	D10	N1	N11	2	D19	N1	N20	3
D2	N1	N3	3	D11	N1	N12	2	D20	N1	N21	3
D3	N1	N4	6	D12	N1	N13	2	D21	N1	N22	5
D4	N1	N5	4	D13	N1	N14	2	D22	N1	N23	8
D5	N1	N6	2	D14	N1	N15	2	D23	N2	N3	4
D6	N1	N7	2	D15	N1	N16	2	D24	N2	N4	6
D7	N1	N8	2	D16	N1	N17	2	D25	N2	N5	3
D8	N1	N9	2	D17	N1	N18	3	D26	N2	N6	2
D9	N1	N10	2	D18	N1	N19	3	D27	N2	N7	2

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D28	N2	N8	2	D70	N4	N11	2	D112	N6	N18	4
D29	N2	N9	1	D71	N4	N12	3	D113	N6	N19	4
D30	N2	N10	1	D72	N4	N13	3	D114	N6	N20	3
D31	N2	N11	1	D73	N4	N14	3	D115	N6	N21	2
D32	N2	N12	2	D74	N4	N15	4	D116	N6	N22	3
D33	N2	N13	2	D75	N4	N16	3	D117	N6	N23	3
D34	N2	N14	2	D76	N4	N17	3	D118	N7	N8	9
D35	N2	N15	2	D77	N4	N18	6	D119	N7	N9	8
D36	N2	N16	1	D78	N4	N19	6	D120	N7	N10	7
D37	N2	N17	2	D79	N4	N20	5	D121	N7	N11	4
D38	N2	N18	3	D80	N4	N21	3	D122	N7	N12	5
D39	N2	N19	2	D81	N4	N22	5	D123	N7	N13	4
D40	N2	N20	2	D82	N4	N23	10	D124	N7	N14	5
D41	N2	N21	2	D83	N5	N6	5	D125	N7	N15	6
D42	N2	N22	3	D84	N5	N7	5	D126	N7	N16	3
D43	N2	N23	4	D85	N5	N8	5	D127	N7	N17	3
D44	N3	N4	8	D86	N5	N9	3	D128	N7	N18	5
D45	N3	N5	5	D87	N5	N10	3	D129	N7	N19	4
D46	N3	N6	5	D88	N5	N11	3	D130	N7	N20	3
D47	N3	N7	4	D89	N5	N12	3	D131	N7	N21	2
D48	N3	N8	3	D90	N5	N13	3	D132	N7	N22	3
D49	N3	N9	2	D91	N5	N14	4	D133	N7	N23	3
D50	N3	N10	2	D92	N5	N15	5	D134	N8	N9	4
D51	N3	N11	2	D93	N5	N16	3	D135	N8	N10	5
D52	N3	N12	2	D94	N5	N17	3	D136	N8	N11	3
D53	N3	N13	2	D95	N5	N18	7	D137	N8	N12	4
D54	N3	N14	3	D96	N5	N19	10	D138	N8	N13	3
D55	N3	N15	3	D97	N5	N20	5	D139	N8	N14	5
D56	N3	N16	2	D98	N5	N21	3	D140	N8	N15	7
D57	N3	N17	2	D99	N5	N22	4	D141	N8	N16	3
D58	N3	N18	4	D100	N5	N23	6	D142	N8	N17	2
D59	N3	N19	3	D101	N6	N7	7	D143	N8	N18	5
D60	N3	N20	3	D102	N6	N8	7	D144	N8	N19	4
D61	N3	N21	2	D103	N6	N9	3	D145	N8	N20	3
D62	N3	N22	3	D104	N6	N10	3	D146	N8	N21	2
D63	N3	N23	4	D105	N6	N11	2	D147	N8	N22	3
D64	N4	N5	10	D106	N6	N12	3	D148	N8	N23	3
D65	N4	N6	5	D107	N6	N13	3	D149	N9	N10	7
D66	N4	N7	4	D108	N6	N14	3	D150	N9	N11	4
D67	N4	N8	4	D109	N6	N15	4	D151	N9	N12	4
D68	N4	N9	2	D110	N6	N16	2	D152	N9	N13	3
D69	N4	N10	3	D111	N6	N17	2	D153	N9	N14	4

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D154	N9	N15	4	D188	N12	N13	16	D222	N15	N20	4
D155	N9	N16	2	D189	N12	N14	10	D223	N15	N21	3
D156	N9	N17	2	D190	N12	N15	5	D224	N15	N22	3
D157	N9	N18	3	D191	N12	N16	5	D225	N15	N23	3
D158	N9	N19	2	D192	N12	N17	3	D226	N16	N17	4
D159	N9	N20	2	D193	N12	N18	5	D227	N16	N18	8
D160	N9	N21	2	D194	N12	N19	3	D228	N16	N19	3
D161	N9	N22	2	D195	N12	N20	3	D229	N16	N20	3
D162	N9	N23	2	D196	N12	N21	2	D230	N16	N21	2
D163	N10	N11	6	D197	N12	N22	3	D231	N16	N22	2
D164	N10	N12	6	D198	N12	N23	3	D232	N16	N23	2
D165	N10	N13	4	D199	N13	N14	11	D233	N17	N18	8
D166	N10	N14	6	D200	N13	N15	5	D234	N17	N19	4
D167	N10	N15	6	D201	N13	N16	6	D235	N17	N20	6
D168	N10	N16	3	D202	N13	N17	4	D236	N17	N21	4
D169	N10	N17	2	D203	N13	N18	6	D237	N17	N22	3
D170	N10	N18	4	D204	N13	N19	3	D238	N17	N23	3
D171	N10	N19	3	D205	N13	N20	4	D239	N18	N19	11
D172	N10	N20	3	D206	N13	N21	3	D240	N18	N20	10
D173	N10	N21	2	D207	N13	N22	3	D241	N18	N21	5
D174	N10	N22	2	D208	N13	N23	3	D242	N18	N22	5
D175	N10	N23	2	D209	N14	N15	9	D243	N18	N23	6
D176	N11	N12	13	D210	N14	N16	8	D244	N19	N20	7
D177	N11	N13	7	D211	N14	N17	4	D245	N19	N21	3
D178	N11	N14	8	D212	N14	N18	8	D246	N19	N22	4
D179	N11	N15	4	D213	N14	N19	4	D247	N19	N23	5
D180	N11	N16	3	D214	N14	N20	4	D248	N20	N21	9
D181	N11	N17	2	D215	N14	N21	3	D249	N20	N22	8
D182	N11	N18	4	D216	N14	N22	3	D250	N20	N23	7
D183	N11	N19	2	D217	N14	N23	3	D251	N21	N22	8
D184	N11	N20	2	D218	N15	N16	5	D252	N21	N23	4
D185	N11	N21	2	D219	N15	N17	3	D253	N22	N23	10
D186	N11	N22	2	D220	N15	N18	9				
D187	N11	N23	2	D221	N15	N19	5				

NETWORK 20n88s

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	10	D41	N3	N7	2	D81	N5	N16	3
D2	N1	N3	3	D42	N3	N8	1	D82	N5	N17	3
D3	N1	N4	3	D43	N3	N9	2	D83	N5	N18	4
D4	N1	N5	3	D44	N3	N10	2	D84	N5	N19	3
D5	N1	N6	3	D45	N3	N11	1	D85	N5	N20	2
D6	N1	N7	2	D46	N3	N12	1	D86	N6	N7	8
D7	N1	N8	2	D47	N3	N13	2	D87	N6	N8	3
D8	N1	N9	2	D48	N3	N14	2	D88	N6	N9	4
D9	N1	N10	2	D49	N3	N15	2	D89	N6	N10	3
D10	N1	N11	1	D50	N3	N16	2	D90	N6	N11	2
D11	N1	N12	1	D51	N3	N17	2	D91	N6	N12	2
D12	N1	N13	2	D52	N3	N18	4	D92	N6	N13	3
D13	N1	N14	2	D53	N3	N19	2	D93	N6	N14	3
D14	N1	N15	2	D54	N3	N20	2	D94	N6	N15	2
D15	N1	N16	2	D55	N4	N5	4	D95	N6	N16	2
D16	N1	N17	2	D56	N4	N6	4	D96	N6	N17	2
D17	N1	N18	4	D57	N4	N7	3	D97	N6	N18	3
D18	N1	N19	3	D58	N4	N8	2	D98	N6	N19	2
D19	N1	N20	3	D59	N4	N9	2	D99	N6	N20	2
D20	N2	N3	5	D60	N4	N10	2	D100	N7	N8	5
D21	N2	N4	4	D61	N4	N11	1	D101	N7	N9	6
D22	N2	N5	5	D62	N4	N12	1	D102	N7	N10	4
D23	N2	N6	3	D63	N4	N13	2	D103	N7	N11	2
D24	N2	N7	3	D64	N4	N14	2	D104	N7	N12	2
D25	N2	N8	2	D65	N4	N15	2	D105	N7	N13	3
D26	N2	N9	3	D66	N4	N16	2	D106	N7	N14	3
D27	N2	N10	3	D67	N4	N17	2	D107	N7	N15	2
D28	N2	N11	2	D68	N4	N18	3	D108	N7	N16	2
D29	N2	N12	2	D69	N4	N19	2	D109	N7	N17	2
D30	N2	N13	3	D70	N4	N20	2	D110	N7	N18	3
D31	N2	N14	3	D71	N5	N6	5	D111	N7	N19	2
D32	N2	N15	3	D72	N5	N7	5	D112	N7	N20	2
D33	N2	N16	3	D73	N5	N8	3	D113	N8	N9	5
D34	N2	N17	3	D74	N5	N9	4	D114	N8	N10	3
D35	N2	N18	7	D75	N5	N10	4	D115	N8	N11	2
D36	N2	N19	4	D76	N5	N11	2	D116	N8	N12	2
D37	N2	N20	6	D77	N5	N12	2	D117	N8	N13	2
D38	N3	N4	2	D78	N5	N13	4	D118	N8	N14	2
D39	N3	N5	3	D79	N5	N14	5	D119	N8	N15	2
D40	N3	N6	2	D80	N5	N15	3	D120	N8	N16	1

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D121	N8	N17	1	D145	N10	N20	2	D169	N13	N20	2
D122	N8	N18	2	D146	N11	N12	3	D170	N14	N15	6
D123	N8	N19	1	D147	N11	N13	3	D171	N14	N16	3
D124	N8	N20	1	D148	N11	N14	2	D172	N14	N17	4
D125	N9	N10	7	D149	N11	N15	2	D173	N14	N18	4
D126	N9	N11	4	D150	N11	N16	1	D174	N14	N19	3
D127	N9	N12	3	D151	N11	N17	1	D175	N14	N20	2
D128	N9	N13	4	D152	N11	N18	2	D176	N15	N16	5
D129	N9	N14	4	D153	N11	N19	1	D177	N15	N17	5
D130	N9	N15	3	D154	N11	N20	1	D178	N15	N18	4
D131	N9	N16	2	D155	N12	N13	5	D179	N15	N19	3
D132	N9	N17	2	D156	N12	N14	3	D180	N15	N20	2
D133	N9	N18	3	D157	N12	N15	2	D181	N16	N17	5
D134	N9	N19	2	D158	N12	N16	2	D182	N16	N18	5
D135	N9	N20	2	D159	N12	N17	2	D183	N16	N19	6
D136	N10	N11	3	D160	N12	N18	2	D184	N16	N20	3
D137	N10	N12	4	D161	N12	N19	1	D185	N17	N18	7
D138	N10	N13	6	D162	N12	N20	1	D186	N17	N19	3
D139	N10	N14	5	D163	N13	N14	6	D187	N17	N20	2
D140	N10	N15	3	D164	N13	N15	6	D188	N18	N19	6
D141	N10	N16	2	D165	N13	N16	3	D189	N18	N20	6
D142	N10	N17	2	D166	N13	N17	3	D190	N19	N20	4
D143	N10	N18	3	D167	N13	N18	4				
D144	N10	N19	2	D168	N13	N19	3				

NETWORK 15n59s

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	2	D13	N1	N14	2	D25	N2	N13	1
D2	N1	N3	6	D14	N1	N15	4	D26	N2	N14	4
D3	N1	N4	4	D15	N2	N3	2	D27	N2	N15	2
D4	N1	N5	2	D16	N2	N4	5	D28	N3	N4	7
D5	N1	N6	2	D17	N2	N5	2	D29	N3	N5	2
D6	N1	N7	4	D18	N2	N6	3	D30	N3	N6	2
D7	N1	N8	2	D19	N2	N7	3	D31	N3	N7	6
D8	N1	N9	3	D20	N2	N8	2	D32	N3	N8	2
D9	N1	N10	2	D21	N2	N9	3	D33	N3	N9	4
D10	N1	N11	1	D22	N2	N10	2	D34	N3	N10	3
D11	N1	N12	1	D23	N2	N11	1	D35	N3	N11	2
D12	N1	N13	1	D24	N2	N12	1	D36	N3	N12	1

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D37	N3	N13	1	D60	N5	N15	3	D83	N8	N14	3
D38	N3	N14	2	D61	N6	N7	4	D84	N8	N15	2
D39	N3	N15	5	D62	N6	N8	6	D85	N9	N10	9
D40	N4	N5	3	D63	N6	N9	4	D86	N9	N11	4
D41	N4	N6	4	D64	N6	N10	2	D87	N9	N12	3
D42	N4	N7	9	D65	N6	N11	2	D88	N9	N13	4
D43	N4	N8	4	D66	N6	N12	2	D89	N9	N14	3
D44	N4	N9	6	D67	N6	N13	2	D90	N9	N15	6
D45	N4	N10	4	D68	N6	N14	6	D91	N10	N11	5
D46	N4	N11	2	D69	N6	N15	2	D92	N10	N12	2
D47	N4	N12	2	D70	N7	N8	4	D93	N10	N13	3
D48	N4	N13	2	D71	N7	N9	17	D94	N10	N14	2
D49	N4	N14	5	D72	N7	N10	7	D95	N10	N15	6
D50	N4	N15	5	D73	N7	N11	3	D96	N11	N12	1
D51	N5	N6	3	D74	N7	N12	2	D97	N11	N13	2
D52	N5	N7	5	D75	N7	N13	3	D98	N11	N14	1
D53	N5	N8	6	D76	N7	N14	4	D99	N11	N15	3
D54	N5	N9	7	D77	N7	N15	7	D100	N12	N13	2
D55	N5	N10	3	D78	N8	N9	5	D101	N12	N14	2
D56	N5	N11	2	D79	N8	N10	3	D102	N12	N15	1
D57	N5	N12	4	D80	N8	N11	2	D103	N13	N14	1
D58	N5	N13	3	D81	N8	N12	4	D104	N13	N15	2
D59	N5	N14	2	D82	N8	N13	2	D105	N14	N15	2

NETWORK 11n55s1

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	9	D15	N2	N7	4	D29	N4	N6	10
D2	N1	N3	6	D16	N2	N8	4	D30	N4	N7	6
D3	N1	N4	4	D17	N2	N9	9	D31	N4	N8	4
D4	N1	N5	3	D18	N2	N10	4	D32	N4	N9	6
D5	N1	N6	4	D19	N2	N11	6	D33	N4	N10	3
D6	N1	N7	3	D20	N3	N4	8	D34	N4	N11	4
D7	N1	N8	3	D21	N3	N5	5	D35	N5	N6	6
D8	N1	N9	5	D22	N3	N6	5	D36	N5	N7	7
D9	N1	N10	4	D23	N3	N7	4	D37	N5	N8	4
D10	N1	N11	5	D24	N3	N8	3	D38	N5	N9	4
D11	N2	N3	7	D25	N3	N9	4	D39	N5	N10	3
D12	N2	N4	7	D26	N3	N10	3	D40	N5	N11	3
D13	N2	N5	4	D27	N3	N11	3	D41	N6	N7	7
D14	N2	N6	7	D28	N4	N5	9	D42	N6	N8	7

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D43	N6	N9	9	D48	N7	N10	3	D53	N9	N10	7
D44	N6	N10	4	D49	N7	N11	3	D54	N9	N11	7
D45	N6	N11	4	D50	N8	N9	8	D55	N10	N11	8
D46	N7	N8	5	D51	N8	N10	7				
D47	N7	N9	4	D52	N8	N11	4				

NETWORK COST239-11nodes

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D01	N0	N1	5	D20	N2	N3	1	D39	N4	N9	2
D02	N0	N2	6	D21	N2	N4	3	D40	N4	N10	1
D03	N0	N3	1	D22	N2	N5	11	D41	N5	N6	8
D04	N0	N4	2	D23	N2	N6	3	D42	N5	N7	2
D05	N0	N5	11	D24	N2	N7	1	D43	N5	N8	6
D06	N0	N6	5	D25	N2	N8	6	D44	N5	N9	8
D07	N0	N7	1	D26	N2	N9	3	D45	N5	N10	3
D08	N0	N8	7	D27	N2	N10	1	D46	N6	N7	1
D09	N0	N9	10	D28	N3	N4	1	D47	N6	N8	4
D10	N0	N10	1	D29	N3	N5	2	D48	N6	N9	5
D11	N1	N2	6	D30	N3	N6	1	D49	N6	N10	1
D12	N1	N3	1	D31	N3	N7	1	D50	N7	N8	1
D13	N1	N4	3	D32	N3	N8	1	D51	N7	N9	1
D14	N1	N5	9	D33	N3	N9	1	D52	N7	N10	1
D15	N1	N6	2	D34	N3	N10	1	D53	N8	N9	4
D16	N1	N7	1	D35	N4	N5	9	D54	N8	N10	1
D17	N1	N8	2	D36	N4	N6	1	D55	N9	N10	1
D18	N1	N9	3	D37	N4	N7	1				
D19	N1	N10	1	D38	N4	N8	1				

NETWORK 10n45s1

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	4	D10	N2	N3	3	D19	N3	N5	2
D2	N1	N3	6	D11	N2	N4	4	D20	N3	N6	2
D3	N1	N4	5	D12	N2	N5	3	D21	N3	N7	3
D4	N1	N5	3	D13	N2	N6	7	D22	N3	N8	6
D5	N1	N6	3	D14	N2	N7	2	D23	N3	N9	2
D6	N1	N7	3	D15	N2	N8	2	D24	N3	N10	4
D7	N1	N8	3	D16	N2	N9	4	D25	N4	N5	5
D8	N1	N9	3	D17	N2	N10	2	D26	N4	N6	3
D9	N1	N10	3	D18	N3	N4	4	D27	N4	N7	4

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D28	N4	N8	3	D34	N5	N9	5	D40	N7	N8	4
D29	N4	N9	3	D35	N5	N10	2	D41	N7	N9	2
D30	N4	N10	3	D36	N6	N7	2	D42	N7	N10	5
D31	N5	N6	3	D37	N6	N8	2	D43	N8	N9	2
D32	N5	N7	3	D38	N6	N9	5	D44	N8	N10	6
D33	N5	N8	2	D39	N6	N10	2	D45	N9	N10	2

NETWORK 10n45s2

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	8	D16	N2	N9	3	D31	N5	N6	6
D2	N1	N3	4	D17	N2	N10	5	D32	N5	N7	8
D3	N1	N4	4	D18	N3	N4	8	D33	N5	N8	4
D4	N1	N5	6	D19	N3	N5	5	D34	N5	N9	4
D5	N1	N6	3	D20	N3	N6	4	D35	N5	N10	6
D6	N1	N7	4	D21	N3	N7	3	D36	N6	N7	5
D7	N1	N8	3	D22	N3	N8	3	D37	N6	N8	5
D8	N1	N9	3	D23	N3	N9	2	D38	N6	N9	3
D9	N1	N10	8	D24	N3	N10	3	D39	N6	N10	3
D10	N2	N3	6	D25	N4	N5	8	D40	N7	N8	8
D11	N2	N4	6	D26	N4	N6	8	D41	N7	N9	6
D12	N2	N5	6	D27	N4	N7	5	D42	N7	N10	6
D13	N2	N6	4	D28	N4	N8	4	D43	N8	N9	5
D14	N2	N7	4	D29	N4	N9	3	D44	N8	N10	4
D15	N2	N8	3	D30	N4	N10	4	D45	N9	N10	5

NETWORK COST239-10nodes

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N0	N1	5	D11	N1	N3	1	D21	N2	N6	3
D2	N0	N2	6	D12	N1	N4	3	D22	N2	N7	1
D3	N0	N3	1	D13	N1	N5	9	D23	N2	N8	6
D4	N0	N4	2	D14	N1	N6	2	D24	N2	N9	3
D5	N0	N5	11	D15	N1	N7	1	D25	N3	N4	1
D6	N0	N6	5	D16	N1	N8	2	D26	N3	N5	2
D7	N0	N7	1	D17	N1	N9	3	D27	N3	N6	1
D8	N0	N8	7	D18	N2	N3	1	D28	N3	N7	1
D9	N0	N9	10	D19	N2	N4	3	D29	N3	N8	1
D10	N1	N2	6	D20	N2	N5	11	D30	N3	N9	1

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D31	N4	N5	9	D36	N5	N6	8	D41	N6	N8	4
D32	N4	N6	1	D37	N5	N7	2	D42	N6	N9	5
D33	N4	N7	1	D38	N5	N8	6	D43	N7	N8	1
D34	N4	N8	1	D39	N5	N9	8	D44	N7	N9	1
D35	N4	N9	2	D40	N6	N7	1	D45	N8	N9	4

NETWORK COST239-9nodes

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	6	D13	N2	N7	1	D25	N4	N8	1
D2	N1	N3	1	D14	N2	N8	6	D26	N4	N9	2
D3	N1	N4	3	D15	N2	N9	3	D27	N5	N6	8
D4	N1	N5	9	D16	N3	N4	1	D28	N5	N7	2
D5	N1	N6	2	D17	N3	N5	2	D29	N5	N8	6
D6	N1	N7	1	D18	N3	N6	1	D30	N5	N9	8
D7	N1	N8	2	D19	N3	N7	1	D31	N6	N7	1
D8	N1	N9	3	D20	N3	N8	1	D32	N6	N8	4
D9	N2	N3	1	D21	N3	N9	1	D33	N6	N9	5
D10	N2	N4	3	D22	N4	N5	9	D34	N7	N8	1
D11	N2	N5	11	D23	N4	N6	1	D35	N7	N9	1
D12	N2	N6	3	D24	N4	N7	1	D36	N8	N9	4

NETWORK 9n36s1

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	9	D13	N2	N7	4	D25	N4	N8	5
D2	N1	N3	11	D14	N2	N8	4	D26	N4	N9	5
D3	N1	N4	6	D15	N2	N9	6	D27	N5	N6	12
D4	N1	N5	5	D16	N3	N4	12	D28	N5	N7	8
D5	N1	N6	6	D17	N3	N5	8	D29	N5	N8	5
D6	N1	N7	4	D18	N3	N6	12	D30	N5	N9	5
D7	N1	N8	6	D19	N3	N7	6	D31	N6	N7	12
D8	N1	N9	11	D20	N3	N8	7	D32	N6	N8	9
D9	N2	N3	11	D21	N3	N9	9	D33	N6	N9	8
D10	N2	N4	11	D22	N4	N5	12	D34	N7	N8	8
D11	N2	N5	6	D23	N4	N6	8	D35	N7	N9	6
D12	N2	N6	6	D24	N4	N7	6	D36	N8	N9	12

NETWORK 9n36s2

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	9	D13	N2	N7	4	D25	N4	N8	5
D2	N1	N3	5	D14	N2	N8	4	D26	N4	N9	5
D3	N1	N4	6	D15	N2	N9	5	D27	N5	N6	12
D4	N1	N5	12	D16	N3	N4	8	D28	N5	N7	6
D5	N1	N6	6	D17	N3	N5	5	D29	N5	N8	8
D6	N1	N7	4	D18	N3	N6	5	D30	N5	N9	11
D7	N1	N8	6	D19	N3	N7	4	D31	N6	N7	11
D8	N1	N9	11	D20	N3	N8	4	D32	N6	N8	9
D9	N2	N3	8	D21	N3	N9	4	D33	N6	N9	7
D10	N2	N4	7	D22	N4	N5	8	D34	N7	N8	11
D11	N2	N5	8	D23	N4	N6	12	D35	N7	N9	5
D12	N2	N6	6	D24	N4	N7	6	D36	N8	N9	9

NETWORK COST239-8nodes

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	6	D11	N2	N6	3	D21	N4	N7	1
D2	N1	N3	1	D12	N2	N7	1	D22	N4	N8	1
D3	N1	N4	3	D13	N2	N8	6	D23	N5	N6	8
D4	N1	N5	9	D14	N3	N4	1	D24	N5	N7	2
D5	N1	N6	2	D15	N3	N5	2	D25	N5	N8	6
D6	N1	N7	1	D16	N3	N6	1	D26	N6	N7	1
D7	N1	N8	2	D17	N3	N7	1	D27	N6	N8	4
D8	N2	N3	1	D18	N3	N8	1	D28	N7	N8	1
D9	N2	N4	3	D19	N4	N5	9				
D10	N2	N5	11	D20	N4	N6	1				

NETWORK 8n28s1 - Gravity

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	10	D11	N2	N6	7	D21	N4	N7	12
D2	N1	N3	9	D12	N2	N7	7	D22	N4	N8	6
D3	N1	N4	7	D13	N2	N8	6	D23	N5	N6	6
D4	N1	N5	8	D14	N3	N4	6	D24	N5	N7	9
D5	N1	N6	20	D15	N3	N5	13	D25	N5	N8	13
D6	N1	N7	17	D16	N3	N6	7	D26	N6	N7	17
D7	N1	N8	6	D17	N3	N7	8	D27	N6	N8	5
D8	N2	N3	10	D18	N3	N8	12	D28	N7	N8	6
D9	N2	N4	5	D19	N4	N5	9				
D10	N2	N5	6	D20	N4	N6	8				

NETWORK 8n28s1 - Random

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	7	D11	N2	N6	6	D21	N4	N7	1
D2	N1	N3	1	D12	N2	N7	9	D22	N4	N8	5
D3	N1	N4	4	D13	N2	N8	7	D23	N5	N6	4
D4	N1	N5	4	D14	N3	N4	4	D24	N5	N7	7
D5	N1	N6	7	D15	N3	N5	5	D25	N5	N8	6
D6	N1	N7	6	D16	N3	N6	8	D26	N6	N7	7
D7	N1	N8	8	D17	N3	N7	7	D27	N6	N8	4
D8	N2	N3	8	D18	N3	N8	8	D28	N7	N8	9
D9	N2	N4	7	D19	N4	N5	4				
D10	N2	N5	2	D20	N4	N6	2				

NETWORK 8n28s2 - Gravity

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	7	D11	N2	N6	3	D21	N4	N7	4
D2	N1	N3	9	D12	N2	N7	4	D22	N4	N8	8
D3	N1	N4	5	D13	N2	N8	4	D23	N5	N6	4
D4	N1	N5	8	D14	N3	N4	4	D24	N5	N7	4
D5	N1	N6	6	D15	N3	N5	8	D25	N5	N8	3
D6	N1	N7	7	D16	N3	N6	8	D26	N6	N7	20
D7	N1	N8	4	D17	N3	N7	7	D27	N6	N8	4
D8	N2	N3	5	D18	N3	N8	3	D28	N7	N8	4
D9	N2	N4	6	D19	N4	N5	4				
D10	N2	N5	7	D20	N4	N6	3				

NETWORK 8n28s2 - Random

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	15	D11	N2	N6	2	D21	N4	N7	6
D2	N1	N3	7	D12	N2	N7	14	D22	N4	N8	15
D3	N1	N4	3	D13	N2	N8	4	D23	N5	N6	9
D4	N1	N5	7	D14	N3	N4	4	D24	N5	N7	3
D5	N1	N6	9	D15	N3	N5	4	D25	N5	N8	2
D6	N1	N7	4	D16	N3	N6	11	D26	N6	N7	12
D7	N1	N8	2	D17	N3	N7	5	D27	N6	N8	8
D8	N2	N3	8	D18	N3	N8	12	D28	N7	N8	1
D9	N2	N4	15	D19	N4	N5	12				
D10	N2	N5	5	D20	N4	N6	1				

NETWORK COST239-7nodes

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	6	D8	N2	N4	3	D15	N3	N7	1
D2	N1	N3	1	D9	N2	N5	11	D16	N4	N5	9
D3	N1	N4	3	D10	N2	N6	3	D17	N4	N6	1
D4	N1	N5	9	D11	N2	N7	1	D18	N4	N7	1
D5	N1	N6	2	D12	N3	N4	1	D19	N5	N6	8
D6	N1	N7	1	D13	N3	N5	2	D20	N5	N7	2
D7	N2	N3	1	D14	N3	N6	1	D21	N6	N7	1

NETWORK 7n21s1 - Gravity

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	7	D8	N2	N4	6	D15	N3	N7	10
D2	N1	N3	5	D9	N2	N5	12	D16	N4	N5	4
D3	N1	N4	10	D10	N2	N6	5	D17	N4	N6	4
D4	N1	N5	5	D11	N2	N7	5	D18	N4	N7	8
D5	N1	N6	3	D12	N3	N4	6	D19	N5	N6	5
D6	N1	N7	5	D13	N3	N5	5	D20	N5	N7	4
D7	N2	N3	7	D14	N3	N6	8	D21	N6	N7	5

NETWORK 7n21s1 - Random

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	3	D8	N2	N4	6	D15	N3	N7	3
D2	N1	N3	4	D9	N2	N5	6	D16	N4	N5	8
D3	N1	N4	3	D10	N2	N6	3	D17	N4	N6	9
D4	N1	N5	6	D11	N2	N7	9	D18	N4	N7	7
D5	N1	N6	9	D12	N3	N4	2	D19	N5	N6	2
D6	N1	N7	10	D13	N3	N5	4	D20	N5	N7	4
D7	N2	N3	7	D14	N3	N6	1	D21	N6	N7	9

NETWORK 7n21s2 - Gravity

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	9	D5	N1	N6	15	D9	N2	N5	5
D2	N1	N3	5	D6	N1	N7	8	D10	N2	N6	7
D3	N1	N4	6	D7	N2	N3	6	D11	N2	N7	5
D4	N1	N5	5	D8	N2	N4	4	D12	N3	N4	4

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D13	N3	N5	9	D16	N4	N5	6	D19	N5	N6	7
D14	N3	N6	6	D17	N4	N6	9	D20	N5	N7	11
D15	N3	N7	6	D18	N4	N7	11	D21	N6	N7	17

NETWORK 7n21s2 - Random

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	5	D8	N2	N4	5	D15	N3	N7	9
D2	N1	N3	9	D9	N2	N5	10	D16	N4	N5	4
D3	N1	N4	5	D10	N2	N6	9	D17	N4	N6	2
D4	N1	N5	9	D11	N2	N7	6	D18	N4	N7	3
D5	N1	N6	10	D12	N3	N4	6	D19	N5	N6	1
D6	N1	N7	9	D13	N3	N5	7	D20	N5	N7	2
D7	N2	N3	10	D14	N3	N6	7	D21	N6	N7	1

NETWORK COST239-6nodes

D	O	D	QTY.	D	O	D	QTY.	D	O	D	QTY.
D1	N1	N2	6	D6	N2	N3	1	D11	N3	N5	2
D2	N1	N3	1	D7	N2	N4	3	D12	N3	N6	1
D3	N1	N4	3	D8	N2	N5	11	D13	N4	N5	9
D4	N1	N5	9	D9	N2	N6	3	D14	N4	N6	1
D5	N1	N6	2	D10	N3	N4	1	D15	N5	N6	8

APPENDIX C

AMPL MODELS

Mesh Topology & Capacity Design (MTRS) using flow variables - John Doucette @ TRLabs, Edmonton

This AMPL model performs optimal joint capacity placement and topology determination. Cost is calculated as the sum of the fixed costs of using each span ("fixed charge" part) and the cost of all units of working and spare capacity placed ("routing" part). The problem is viewed as a pure MCMF type flow solution using source-sink and "transshipment" type constraints for each demand relation.

Note that while the affected transportation flow-like expression is computationally advantageous in removing O(all pairs) sets of explicit eligible route representations, it strictly implies that there is no definite hop or distance limit on working route lengths. This can however, be inspected or assessed by experiment on the resulting design or by inspection of individual restoration flow quantities for each relation. The added constraint "max_span" restricts number of spans to be no more than needed for degree 5 network.

```
# TOPOLOGY DEFINITION
set SPANS;
    # Set of all logical spans
set NODES;
    # Set of all logical nodes
set DEMANDS;
    # Set of all demands
param Incidence{j in SPANS, n in NODES} default 0;
    # Equal to 1 if span j is incident on node n, 0 otherwise
param IncidenceA{j in SPANS, n in NODES} default 0;
    # Equal to 1 if span j starts at node n, 0 otherwise (abitrary whether starts or ends on node)
param IncidenceB{j in SPANS, n in NODES} default 0;
    # Equal to 1 if span j ends at node n, 0 otherwise

# WORKING DEMANDS
param Origin{r in DEMANDS} symbolic in NODES;
    # Origin node of span demand r
param Destination{r in DEMANDS} symbolic in NODES;
    # Destination node of demand r
param DemandUnits{r in DEMANDS} default 0;
    # Size of demand r
# COST DATA
param UnitCost{j in SPANS};
    # The cost of placing a unit of capacity on span j
param FixedCost{j in SPANS};
    # The fixed cost of simply using span j, regardless of the actual capacity placed

# VARIABLES
var work_flow_from{n in NODES, j in SPANS, r in DEMANDS: Incidence[j,n]=1} >=0,
<=DemandUnits[r];
var work_flow_into{n in NODES, j in SPANS, r in DEMANDS: Incidence[j,n]=1} >=0,
<=DemandUnits[r];
    # Total directed working traffic flows from and into node n on span j for demand r.
var spare_flow_from{n in NODES, j in SPANS, i in SPANS: Incidence[j,n]=1 and i<>j} >=0,
<=10000;
var spare_flow_into{n in NODES, j in SPANS, i in SPANS: Incidence[j,n]=1 and i<>j} >=0,
<=10000;
    # Total directed restoration flows from and into node n on span j for span failure i.
var psi{j in SPANS} >=0, <=1 integer;
    # Equal to 1 if span j is used, equal to 0 otherwise
var work{j in SPANS} >=0, <=10000 integer;
    # Total amount of working capacity (i.e. number of wavelengths) placed on span j.
var spare{j in SPANS} >=0, <=10000 integer;
    # Total amount of spare capacity (i.e. number of wavelengths) placed on span j.

# OBJECTIVE FUNCTION
minimize total_cost:
sum{j in SPANS} ( (spare[j] + work[j]) * UnitCost[j] + psi[j] * FixedCost[j] );

# WORK-RELATED CONSTRAINTS
subject to source_work_flows{r in DEMANDS, n in NODES: n = Origin[r]}:
sum {j in SPANS: Incidence[j,n]=1} work_flow_from[n,j,r] = DemandUnits[r];
    # Working flows for demand r flowing FROM the origin node is equal to the total demand
    # on that demand pair.
```

```
subject to no_work_flow_into_origin{r in DEMANDS, n in NODES, j in SPANS: n = Origin[r]
and Incidence[j,n]=1}:
work_flow_into[n,j,r] = 0;
    # Working flows into the origin of a demand are zero.

subject to sink_work_flows{r in DEMANDS, n in NODES: n = Destination[r]}:
sum {j in SPANS: Incidence[j,n]=1} work_flow_into[n,j,r] = DemandUnits[r];
    # Working flows for demand r flowing INTO the destination node is equal to the total demand
    # on that demand pair.

subject to no_work_flow_from_destination{r in DEMANDS, n in NODES, j in SPANS: n =
Destination[r] and Incidence[j,n]=1}:
work_flow_from[n,j,r] = 0;
    # Working flows from the destination of a demand are zero.

subject to work_flow_conservation{r in DEMANDS, n in NODES: n <> Origin[r] and
n<>Destination[r]}:
sum {j in SPANS: Incidence[j,n]=1} work_flow_from[n,j,r] = sum {j in SPANS: Incidence[j,n]=1}
work_flow_into[n,j,r];
    # Flow coming out of a node equals flow going into the node.

subject to anti_symettry_w1{j in SPANS, r in DEMANDS}:
sum {n in NODES: IncidenceA[j,n]=1} work_flow_from[n,j,r] = sum {n in NODES:
IncidenceB[j,n]=1} work_flow_into[n,j,r];
    # Flow from one node on a span must be going into the other node on that same span.

subject to anti_symettry_w2{j in SPANS, r in DEMANDS}:
sum {n in NODES: IncidenceB[j,n]=1} work_flow_from[n,j,r] = sum {n in NODES:
IncidenceA[j,n]=1} work_flow_into[n,j,r];
    # Flow from one node on a span must be going into the other node on that same span.

subject to working_capacity_placement{j in SPANS}:
work[j] >= sum{n in NODES, r in DEMANDS: Incidence[j,n]=1} work_flow_from[n,j,r];
    # Sufficient working capacity must be placed on span j to accomodate all flows routed over it.
# RESTORATION-RELATED CONSTRAINTS
subject to source_spare_flows{i in SPANS, n in NODES: IncidenceA[i,n]=1}:
sum {j in SPANS: Incidence[j,n]=1 and i<>j} spare_flow_from[n,j,i] = work[i];
    # Restoration flows for failure of span i flowing FROM its origin node is equal to the total
    #working capacity on that failed span.

subject to no_spare_flow_into_origin{i in SPANS, n in NODES, j in SPANS: IncidenceA[i,n]=1
and Incidence[j,n]=1 and i<>j}:
spare_flow_into[n,j,i] = 0;
    # Restoration flows into the origin of a failed span are zero.

subject to sink_spare_flows{i in SPANS, n in NODES: IncidenceB[i,n]=1}:
sum {j in SPANS: Incidence[j,n]=1 and i<>j} spare_flow_into[n,j,i] = work[i];
    # Restoration flows for failure of span i flowing INTO the destination node is equal to the total
    #working capacity on that failed span.

subject to no_spare_flow_from_destination{i in SPANS, n in NODES, j in SPANS:
IncidenceB[i,n]=1 and Incidence[j,n]=1 and i<>j}:
spare_flow_from[n,j,i] = 0;
    # Restoration flows from the destination of a failed span are zero.

subject to spare_flow_conservation{i in SPANS, n in NODES: Incidence[i,n]<>1}:
sum {j in SPANS: Incidence[j,n]=1 and i<>j} spare_flow_from[n,j,i] = sum {j in SPANS:
Incidence[j,n]=1 and i<>j} spare_flow_into[n,j,i];
    # Flow coming out of a node equals flow going into the node.

subject to anti_symettry_s1{j in SPANS, i in SPANS: i<>j}:
sum {n in NODES: IncidenceA[j,n]=1} spare_flow_from[n,j,i] = sum {n in NODES:
IncidenceB[j,n]=1} spare_flow_into[n,j,i];
    # Flow from one node on a span must be going into the other node on that same span.

subject to anti_symettry_s2{j in SPANS, i in SPANS: i<>j}:
sum {n in NODES: IncidenceB[j,n]=1} spare_flow_from[n,j,i] = sum {n in NODES:
IncidenceA[j,n]=1} spare_flow_into[n,j,i];
    # Flow from one node on a span must be going into the other node on that same span.
```

```
subject to spare_capacity_placement{j in SPANS, i in SPANS, n in NODES: IncidenceA[j,n]=1 and
i<>j}:
spare[j] >= spare_flow_from[n,j,i] + spare_flow_into[n,j,i];
    # Sufficient spare capacity must be placed on span j to accomodate all flows routed over it
    # for each span failure i.

# SPAN-USE CONSTRAINT
subject to span_use{j in SPANS}:
work[j] + spare[j] <= psi[j]*25000;
    # If there is any capacity placed on span j, then psi[j] must be forced to equal one
    # (i.e. span j is used).

subject to minimum_span_count:
sum{i in SPANS} psi[i] >= card(NODES);
    # There must be a minimum number of spans equal to the total number of nodes for the
    #network to be fully restorable.

subject to nodal_degree{n in NODES}:
sum{i in SPANS: Incidence[i,n]=1} psi[i] >= 2;
    # All nodes must have degree 2 or higher for full restorability.

subject to max_span:
sum{i in SPANS} psi[i] <= 2.5 * card(NODES);
    # There should be no more spans than necessary for a network of mean degree 5.
```

Incremental Topology Growth - Topology & Capacity Design for Network Growth using flow variables - Chioma Ezema @TRLabs

This AMPL model performs optimal joint topology and capacity design (for growth in the network) in which there exists a legacy topology to which we add new growth nodes. New additional spans are chosen and added to the network. Cost is calculated as the sum of the fixed costs of using each new span ("fixed charge" part) and the cost of all units of working and spare capacity placed ("routing" part) on any edge. The *.snif file used should include the currently existing spans and the new spans under consideration.

The problem is viewed as a pure MCMF type flow solution using source-sink and "transshipment" type constraints for each demand relation. Note that while the affected transportation flow-like expression is computationally advantageous in removing O(all pairs) sets of explicit eligible route representations, it strictly implies that there is no definite hop or distance limit on working route lengths. This can however, be inspected or assessed by experiment on the resulting design or by inspection of individual restoration flow quantities for each relation.

```
# TOPOLOGY DEFINITION
set SPANS;
    # Set of all logical spans
set NEW_SPANS within SPANS;
    # Set of new spans to consider for addition
set EXISTING_SPANS within SPANS := {SPANS diff NEW_SPANS};
    # i.e. these are the set of already existing spans that will not be eliminated.
set NODES;
    # Set of all logical nodes (old and new)
set DEMANDS;
    # Set of new demands (involving the new nodes only)
param Incidence{j in SPANS, n in NODES} default 0;
    # Equal to 1 if span j is incident on node n, 0 otherwise
param IncidenceA{j in SPANS, n in NODES} default 0;
    # Equal to 1 if span j starts at node n, 0 otherwise (abitrary whether starts or ends on node)
param IncidenceB{j in SPANS, n in NODES} default 0;
    # Equal to 1 if span j ends at node n, 0 otherwise

# WORKING DEMANDS
param Origin{r in DEMANDS} symbolic in NODES;
```

```
    # Origin node of span demand r
param Destination{r in DEMANDS} symbolic in NODES;
    # Destination node of demand r
param DemandUnits{r in DEMANDS} default 0;
    # Size of demand r

# COST DATA
param UnitCost{j in SPANS};
    # The cost of placing a unit of capacity on span j
param FixedCost{j in NEW_SPANS};
    # The fixed cost of using span j, regardless of the actual capacity placed

# VARIABLES
var work_flow_from{n in NODES, j in SPANS, r in DEMANDS: Incidence[j,n]=1} >=0,
<=DemandUnits[r];
var work_flow_into{n in NODES, j in SPANS, r in DEMANDS: Incidence[j,n]=1} >=0,
<=DemandUnits[r];
    # Total directed working traffic flows from and into node n on span j for demand r
var spare_flow_from{n in NODES, j in SPANS, i in SPANS: Incidence[j,n]=1 and i<>j} >=0,
<=10000;
var spare_flow_into{n in NODES, j in SPANS, i in SPANS: Incidence[j,n]=1 and i<>j} >=0,
<=10000;
    # Total directed restoration flows from and into node n on span j for span failure i
var psi{j in NEW_SPANS} >=0, <=1 integer;
    # Equal to 1 if span j is used, equal to 0 otherwise
var work{j in SPANS} >=0, <=10000 integer;
    # Total amount of working capacity (i.e. number of wavelengths) placed on span j
var spare{j in SPANS} >=0, <=10000 integer;
    # Total amount of spare capacity (i.e. number of wavelengths) placed on span j

# OBJECTIVE FUNCTION
minimize total_additional_cost:
(sum{j in NEW_SPANS} ( (work[j] + spare[j]) * UnitCost[j] + psi[j] * FixedCost[j] )
+ sum{j in EXISTING_SPANS} ( (work[j] + spare[j]) * UnitCost[j] ));
```

```
# WORK-RELATED CONSTRAINTS
subject to source_work_flows{r in DEMANDS, n in NODES: n =  Origin[r]}:
sum {j in SPANS: Incidence[j,n]=1} work_flow_from[n,j,r] = DemandUnits[r];
      # Working flows for demand r flowing FROM the origin node is equal to the total demand
      # on that demand pair

subject to no_work_flow_into_origin{r in DEMANDS, n in NODES, j in SPANS: n =  Origin[r]
and Incidence[j,n]=1}:
work_flow_into[n,j,r] = 0;          # Working flows into the origin of a demand are zero.
subject to sink_work_flows{r in DEMANDS, n in NODES: n = Destination[r]}:
sum {j in SPANS: Incidence[j,n]=1} work_flow_into[n,j,r] = DemandUnits[r];
      # Working flows for demand r flowing INTO the destination node is equal to the total demand
      # on that demand pair.

subject to no_work_flow_from_destination{r in DEMANDS, n in NODES, j in SPANS: n =
Destination[r] and Incidence[j,n]=1}:
work_flow_from[n,j,r] = 0;
      # Working flows from the destination of a demand are zero.

subject to work_flow_conservation{r in DEMANDS, n in NODES: n <> Origin[r] and
n<>Destination[r]}:
sum {j in SPANS: Incidence[j,n]=1} work_flow_from[n,j,r] = sum {j in SPANS: Incidence[j,n]=1}
work_flow_into[n,j,r];
      # Flow coming out of a node equals flow going into the node.

subject to anti_symettry_w1{j in SPANS, r in DEMANDS}:
sum {n in NODES: IncidenceA[j,n]=1} work_flow_from[n,j,r] = sum {n in NODES:
IncidenceB[j,n]=1} work_flow_into[n,j,r];
      # Flow from one node on a span must be going into the other node on that same span

subject to anti_symettry_w2{j in SPANS, r in DEMANDS}:
sum {n in NODES: IncidenceB[j,n]=1} work_flow_from[n,j,r] = sum {n in NODES:
IncidenceA[j,n]=1} work_flow_into[n,j,r];
      # Flow from one node on a span must be going into the other node on that same span
subject to working_capacity_placement{j in SPANS}:
```

```
work[j] >= sum{n in NODES, r in DEMANDS: Incidence[j,n]=1} work_flow_from[n,j,r];
    # Sufficient working capacity must be placed on span j to accommodate all flows routed over it

# RESTORATION-RELATED CONSTRAINTS
subject to source_spare_flows{i in SPANS, n in NODES: IncidenceA[i,n]=1}:
sum {j in SPANS: Incidence[j,n]=1 and i<>j} spare_flow_from[n,j,i] = work[i];
    # Restoration flows for failure of span i flowing FROM its origin node is equal to the total
#working capacity on that failed span.

subject to no_spare_flow_into_origin{i in SPANS, n in NODES, j in SPANS: IncidenceA[i,n]=1
and Incidence[j,n]=1 and i<>j}:
spare_flow_into[n,j,i] = 0;
    # Restoration flows into the origin of a failed span are zero.

subject to sink_spare_flows{i in SPANS, n in NODES: IncidenceB[i,n]=1}:
sum {j in SPANS: Incidence[j,n]=1 and i<>j} spare_flow_into[n,j,i] = work[i];
    # Restoration flows for failure of span i flowing INTO the destination node is equal to the total
#working capacity on that failed span

subject to no_spare_flow_from_destination{i in SPANS, n in NODES, j in SPANS:
IncidenceB[i,n]=1 and Incidence[j,n]=1 and i<>j}:
spare_flow_from[n,j,i] = 0;
    # Restoration flows from the destination of a failed span are zero.

subject to spare_flow_conservation{i in SPANS, n in NODES: Incidence[i,n]<>1}:
sum {j in SPANS: Incidence[j,n]=1 and i<>j} spare_flow_from[n,j,i] = sum {j in SPANS:
Incidence[j,n]=1 and i<>j} spare_flow_into[n,j,i];
    # Flow coming out of a node equals flow going into the node.

subject to anti_symettry_s1{j in SPANS, i in SPANS: i<>j}:
sum {n in NODES: IncidenceA[j,n]=1} spare_flow_from[n,j,i] = sum {n in NODES:
IncidenceB[j,n]=1} spare_flow_into[n,j,i];
    # Flow from one node on a span must be going into the other node on that same span.

subject to anti_symettry_s2{j in SPANS, i in SPANS: i<>j}:
```

```
sum {n in NODES: IncidenceB[j,n]=1} spare_flow_from[n,j,i] = sum {n in NODES:
IncidenceA[j,n]=1} spare_flow_into[n,j,i];
    # Flow from one node on a span must be going into the other node on that same span.

subject to spare_capacity_placement{j in SPANS, i in SPANS, n in NODES: IncidenceA[j,n]=1 and
i<>j}:
spare[j] >= spare_flow_from[n,j,i] + spare_flow_into[n,j,i];
    # Sufficient spare capacity must be placed on span j to accomodate all flows routed over it
    # for each span failure i.

subject to span_use{j in NEW_SPANS}:
work[j] + spare[j] <= psi[j]*25000;
    # If there is any spare capacity placed on span j, then psi[j] must be forced to equal one
    # (i.e. span j is used).

subject to nodal_degree{n in NODES}:
(sum{i in NEW_SPANS: Incidence[i,n]=1} psi[i]) +
(card{i in EXISTING_SPANS: Incidence[i,n]=1}) >= 2;
    # All nodes must have degree 2 or higher for full restorability.
```

Joint Capacity Placement (JCP) – Working and Spare Capacity design using flow variables - Chioma Ezema @TRLabs, Edmonton

This AMPL model performs optimal joint capacity design for the working and spare capacity requirements for the network. Cost is calculated as the sum of all units of working and spare capacity placed on any edge.

The problem is viewed as a pure MCMF type flow solution using source-sink and "transshipment" type constraints for each demand relation. The transportation flow-like expression is computationally advantageous in removing O(all pairs) sets of explicit eligible route representations but it implies that there is no definite hop or distance limit on working route lengths. This can however, be inspected or assessed by experiment on the resulting design or by inspection of individual restoration flow quantities for each relation.

```
# TOPOLOGY DEFINITION
set SPANS;
    # Set of all logical spans
set NODES;
    # Set of all logical nodes
set DEMANDS;
    # Set of all demands
param Incidence{j in SPANS, n in NODES} default 0;
    # Equal to 1 if span j is incident on node n, 0 otherwise
param IncidenceA{j in SPANS, n in NODES} default 0;
    # Equal to 1 if span j starts at node n, 0 otherwise (arbitrary whether starts or ends on node)
param IncidenceB{j in SPANS, n in NODES} default 0;
    # Equal to 1 if span j ends at node n, 0 otherwise

# WORKING DEMANDS
param Origin{r in DEMANDS} symbolic in NODES;
    # Origin node of span demand r
param Destination{r in DEMANDS} symbolic in NODES;
    # Destination node of demand r
param DemandUnits{r in DEMANDS} default 0;
    # Size of demand r
# COST DATA
param UnitCost{j in SPANS};
    # The cost of placing a unit of capacity on span j

# VARIABLES
var work_flow_from{n in NODES, j in SPANS, r in DEMANDS: Incidence[j,n]=1} >=0,
<=DemandUnits[r];
var work_flow_into{n in NODES, j in SPANS, r in DEMANDS: Incidence[j,n]=1} >=0,
<=DemandUnits[r];
    # Total directed working traffic flows from and into node n on span j for demand r
var spare_flow_from{n in NODES, j in SPANS, i in SPANS: Incidence[j,n]=1 and i<>j} >=0,
<=10000;
var spare_flow_into{n in NODES, j in SPANS, i in SPANS: Incidence[j,n]=1 and i<>j} >=0,
<=10000;
    # Total directed restoration flows from and into node n on span j for span failure i
var work{j in SPANS} >=0, <=10000 integer;
    # Total amount of working capacity (i.e. number of wavelengths) placed on span j
var spare{j in SPANS} >=0, <=10000 integer;
    # Total amount of spare capacity (i.e. number of wavelengths) placed on span j

# OBJECTIVE FUNCTION
minimize total_cost:
sum{j in SPANS} ( (spare[j] + work[j]) * UnitCost[j] );

# WORK-RELATED CONSTRAINTS
subject to source_work_flows{r in DEMANDS, n in NODES: n =  Origin[r]}:
sum {j in SPANS: Incidence[j,n]=1} work_flow_from[n,j,r] = DemandUnits[r];
    # Working flows for demand r flowing FROM the origin node is equal to the total demand
    # on that demand pair.

subject to no_work_flow_into_origin{r in DEMANDS, n in NODES, j in SPANS: n =  Origin[r]
and Incidence[j,n]=1}:
work_flow_into[n,j,r] = 0;
    # Working flows into the origin of a demand are zero.
```

```
subject to sink_work_flows{r in DEMANDS, n in NODES: n = Destination[r]}:
sum {j in SPANS: Incidence[j,n]=1} work_flow_into[n,j,r] = DemandUnits[r];
    # Working flows for demand r flowing INTO the destination node is equal to the total demand
    # on that demand pair.

subject to no_work_flow_from_destination{r in DEMANDS, n in NODES, j in SPANS: n =
Destination[r] and Incidence[j,n]=1}:
work_flow_from[n,j,r] = 0;
    # Working flows from the destination of a demand are zero.

subject to work_flow_conservation{r in DEMANDS, n in NODES: n <> Origin[r] and
n<>Destination[r]}:
sum {j in SPANS: Incidence[j,n]=1} work_flow_from[n,j,r] = sum {j in SPANS: Incidence[j,n]=1}
work_flow_into[n,j,r];
    # Flow coming out of a node equals flow going into the node.

subject to anti_symettry_w1{j in SPANS, r in DEMANDS}:
sum {n in NODES: IncidenceA[j,n]=1} work_flow_from[n,j,r] = sum {n in NODES:
IncidenceB[j,n]=1} work_flow_into[n,j,r];
    # Flow from one node on a span must be going into the other node on that same span.

subject to anti_symettry_w2{j in SPANS, r in DEMANDS}:
sum {n in NODES: IncidenceB[j,n]=1} work_flow_from[n,j,r] = sum {n in NODES:
IncidenceA[j,n]=1} work_flow_into[n,j,r];
    # Flow from one node on a span must be going into the other node on that same span.

subject to working_capacity_placement{j in SPANS}:
work[j] >= sum{n in NODES, r in DEMANDS: Incidence[j,n]=1} work_flow_from[n,j,r];
    # Sufficient working capacity must be placed on span j to accomodate all flows routed over it.

# RESTORATION-RELATED CONSTRAINTS
subject to source_spare_flows{i in SPANS, n in NODES: IncidenceA[i,n]=1}:
sum {j in SPANS: Incidence[j,n]=1 and i<>j} spare_flow_from[n,j,i] = work[i];
    # Restoration flows for failure of span i flowing FROM its origin node is equal to the total
working capacity on that failed span.
```

```
subject to no_spare_flow_into_origin{i in SPANS, n in NODES, j in SPANS: IncidenceA[i,n]=1
and Incidence[j,n]=1 and i<>j}:
spare_flow_into[n,j,i] = 0;
    # Restoration flows into the origin of a failed span are zero.

subject to sink_spare_flows{i in SPANS, n in NODES: IncidenceB[i,n]=1}:
sum {j in SPANS: Incidence[j,n]=1 and i<>j} spare_flow_into[n,j,i] = work[i];
    # Restoration flows for failure of span i flowing INTO the destination node is equal to the total
working capacity on that failed span.

subject to no_spare_flow_from_destination{i in SPANS, n in NODES, j in SPANS:
IncidenceB[i,n]=1 and Incidence[j,n]=1 and i<>j}:
spare_flow_from[n,j,i] = 0;
    # Restoration flows from the destination of a failed span are zero.

subject to spare_flow_conservation{i in SPANS, n in NODES: Incidence[i,n]<>1}:
sum {j in SPANS: Incidence[j,n]=1 and i<>j} spare_flow_from[n,j,i] = sum {j in SPANS:
Incidence[j,n]=1 and i<>j} spare_flow_into[n,j,i];
    # Flow coming out of a node equals flow going into the node.

subject to anti_symettry_s1{j in SPANS, i in SPANS: i<>j}:
sum {n in NODES: IncidenceA[j,n]=1} spare_flow_from[n,j,i] = sum {n in NODES:
IncidenceB[j,n]=1} spare_flow_into[n,j,i];
    # Flow from one node on a span must be going into the other node on that same span.

subject to anti_symettry_s2{j in SPANS, i in SPANS: i<>j}:
sum {n in NODES: IncidenceB[j,n]=1} spare_flow_from[n,j,i] = sum {n in NODES:
IncidenceA[j,n]=1} spare_flow_into[n,j,i];
    # Flow from one node on a span must be going into the other node on that same span.

subject to spare_capacity_placement{j in SPANS, i in SPANS, n in NODES: IncidenceA[j,n]=1 and
i<>j}:
spare[j] >= spare_flow_from[n,j,i] + spare_flow_into[n,j,i];
    # Sufficient spare capacity must be placed on span j to accomodate all flows routed over it
    # for each span failure i.
```

Spare Capacity Placement (SCP) using flow variables - Chioma Ezema @TRLabs, Edmonton

This AMPL model performs optimal spare capacity design for the network. Cost is calculated as the sum of all units of spare capacity placed on any edge. The problem is viewed as a pure MCMF type flow solution using source-sink and "transshipment" type constraints for each demand relation. The transportation flow-like expression is computationally advantageous in removing O(all pairs) sets of explicit eligible route representations but it implies that there is no definite hop or distance limit on working route lengths. This can however, be inspected or assessed by experiment on the resulting design or by inspection of individual restoration flow quantities for each relation.

```
# TOPOLOGY DEFINITION
set SPANS;
    # Set of all logical spans
set NODES;
    # Set of all logical nodes

param Incidence{j in SPANS, n in NODES} default 0;
    # Equal to 1 if span j is incident on node n, 0 otherwise
param IncidenceA{j in SPANS, n in NODES} default 0;
    # Equal to 1 if span j starts at node n, 0 otherwise (abitrary whether starts or ends on node)
param IncidenceB{j in SPANS, n in NODES} default 0;
    # Equal to 1 if span j ends at node n, 0 otherwise

# COST DATA
param UnitCost{j in SPANS};
    # The cost of placing a unit of capacity on span j

# WORK DATA
param Work{j in SPANS};
    # Total amount of working capacity (i.e. number of wavelengths) placed on span j.

# VARIABLES
var spare_flow_from{n in NODES, j in SPANS, i in SPANS: Incidence[j,n]=1 and i<>j} >=0,
<=10000;
```

```
var spare_flow_into{n in NODES, j in SPANS, i in SPANS: Incidence[j,n]=1 and i<>j} >=0,
<=10000;
    # Total directed restoration flows from and into node n on span j for span failure i.
var spare{j in SPANS} >=0, <=10000 integer;
    # Total amount of spare capacity (i.e. number of wavelengths) placed on span j.

# OBJECTIVE FUNCTION
minimize total_cost:
sum{j in SPANS} ( spare[j]  * UnitCost[j] );

# RESTORATION-RELATED CONSTRAINTS
subject to source_spare_flows{i in SPANS, n in NODES: IncidenceA[i,n]=1}:
sum {j in SPANS: Incidence[j,n]=1 and i<>j} spare_flow_from[n,j,i] = Work[i];
    # Restoration flows for failure of span i flowing FROM its origin node is equal to the total
working capacity on that failed span.

subject to no_spare_flow_into_origin{i in SPANS, n in NODES, j in SPANS: IncidenceA[i,n]=1
and Incidence[j,n]=1 and i<>j}:
spare_flow_into[n,j,i] = 0;
    # Restoration flows into the origin of a failed span are zero.

subject to sink_spare_flows{i in SPANS, n in NODES: IncidenceB[i,n]=1}:
sum {j in SPANS: Incidence[j,n]=1 and i<>j} spare_flow_into[n,j,i] = Work[i];
    # Restoration flows for failure of span i flowing INTO the destination node is equal to the total
working capacity on that failed span.

subject to no_spare_flow_from_destination{i in SPANS, n in NODES, j in SPANS:
IncidenceB[i,n]=1 and Incidence[j,n]=1 and i<>j}:
spare_flow_from[n,j,i] = 0;
    # Restoration flows from the destination of a failed span are zero.

subject to spare_flow_conservation{i in SPANS, n in NODES: Incidence[i,n]<>1}:
sum {j in SPANS: Incidence[j,n]=1 and i<>j} spare_flow_from[n,j,i] = sum {j in SPANS:
Incidence[j,n]=1 and i<>j} spare_flow_into[n,j,i];
    # Flow coming out of a node equals flow going into the node.
```

```
subject to anti_symettry_s1{j in SPANS, i in SPANS: i<>j}:
sum {n in NODES: IncidenceA[j,n]=1} spare_flow_from[n,j,i] = sum {n in NODES:
IncidenceB[j,n]=1} spare_flow_into[n,j,i];
    # Flow from one node on a span must be going into the other node on that same span.

subject to anti_symettry_s2{j in SPANS, i in SPANS: i<>j}:
sum {n in NODES: IncidenceB[j,n]=1} spare_flow_from[n,j,i] = sum {n in NODES:
IncidenceA[j,n]=1} spare_flow_into[n,j,i];
    # Flow from one node on a span must be going into the other node on that same span.

subject to spare_capacity_placement{j in SPANS, i in SPANS, n in NODES: IncidenceA[j,n]=1 and
i<>j}:
spare[j] >= spare_flow_from[n,j,i] + spare_flow_into[n,j,i];
    # Sufficient spare capacity must be placed on span j to accommodate all flows routed over it for
each span failure i.
```